Drivven Combustion Analysis Toolkit (DCAT)
User's Manual
Version 2.90
July 2010

Drivven, Inc. • 12001 Network Blvd, Bldg E, Suite 110 • San Antonio, Texas 78249 • USA
Phone : 210.248.9308
Web : www.drivven.com , E-mail : info@drivven.com

Table of Contents

Introduction 3
Installing and Activating DCAT 6
Abbreviations 8
DCAT Hardware Setup 9
Encoder Setup and Options 14
TDC Offset Calibration 18
DCAT LabVIEW RT Palette 19
Creating the DCAT Simple Example 20
Reviewing the DCAT Advanced Example 30
  Reviewing the DCATAdvanced Target VI 32
  Reviewing the DCATAdvanced FPGA VI 38
  Reviewing the DCATAdvanced User Interface VI 40
Running the DCATAdvanced Example 41
  Starting DCATAdvanced Example 42
  Measurement and Engine Setup 43
    Measurement Setup 43
    Medium Speed / FPGA DAQ Setup 45
    Engine Setup 47
  Motoring, TDC Alignment and Pegging 49
  Filtering 51
  Viewing All Waveforms 52
    PV Diagram 54
    Next Cycle Control 55
  Calculation Setup and Results 56
    Pressure Metrics 56
    Heat Release 57
    Knock 58
  Viewing Trends and Summary Data 60
    Trends 60
    Summary 61
  Saving Data 62
  Other 64
Overview of the uDCAT Example 65
Creating a Post Processor Application 66
  Opening TDMS Files in Excel 67
  Example Post Processing (Summary Files) 68
  Opening a Raw Data File 70
  Example Post Processing (Raw Files) 72
Case Studies 74
  Inline 4 Engine 75
  V8 Engine 77
Performance 80
  VI Benchmarking 81
  DAQ Hardware 82
  HDD 83
Frequently Asked Questions 84
Appendix A – Toolkit VIs 88
Appendix B – References 229

Introduction

The Drivven Combustion Analysis Toolkit is a unique LabVIEW-based VI toolkit enabling users to integrate sophisticated combustion analysis and logging with engine control applications. The toolkit includes over 75 custom analysis functions, front panel controls and utilities including data streaming to disk, pre-processing, heat release and pressure metrics, summary data reporting and logging, post-processing, and knock and noise analysis. DCAT leverages R-Series (FPGA) and S-Series (simultaneous analog sampling) cards from National Instruments to supervise engine position tracking and synchronization of data collection and processing.

Although DCAT supports a variety of M-Series and S-Series multi-function DAQ devices from National Instruments, the S-Series (6122 and higher) and R-Series card combination can take advantage of the advanced triggering and sampling features within the toolkit. For example, when implemented with S-Series 6122 cards or higher, custom FPGA-based control parameters such as injection duration, fuel pressure, etc., can be stored with each cycle of data. This allows researchers to review data at a later time and identify critical control parameters cycle by cycle.

DCAT allows streaming of raw data to file continuously, by cycle count, time, or file size. Raw data files are stored in National Instrument's popular TDMS format and include all channel configurations, engine geometry and custom test data to allow complete reconstruction of the test environment at a later date. Cycle by cycle summaries of analysis parameters can also be logged to TDMS files for later review. TDMS files may be opened and analyzed in Excel using a free TDMS file plug-in.

The DCAT example application may be used stand-alone, or integrated with an engine control application. DCAT users may insert additional custom analysis and decision blocks for unique test and control applications. A DCAT post-processing application is also included which utilizes the same toolkit.

DCAT Features

Measurement Configuration and Engine Geometry

- Supports multiple S-Series Cards for simultaneous sampling of signals in both engine-synchronous (cylinder pressure, etc.) and engine-asynchronous (knock, etc.) domains
- Individual channel scaling features
  - Cylinder Assignment
  - Gain
  - Gain + Offset
  - Polynomial
  - Table Lookup
- Channels can be assigned to all cylinders as references such as pegging
- Complete set of engine geometry inputs for cylinder volume calculation
  - Optionally load volume from file
- Online motoring test assists with engine geometry verification

Engine Position Tracking and Sample Clock Generation

- Use traditional optical encoder setup or advanced triggering options via R-Series FPGA card
- Flexible FPGA-based crankshaft encoder signal processing and filtering
- Allows online adjustment of TDC index
- Allows use of pressure signal for 4-stroke cam phase information
- Allows cam phase signal to be extended for use as half-cycle gate for TDC index
- Engine Position Tracking (EPT) block allows extrapolation of lower resolution production trigger wheels for high-resolution sampling and control

Multiple Time Based Measurement

- Engine Synchronous Measurements (Sampled at fixed crank angles)
- Engine Asynchronous Measurements (Sampled at a fixed frequency)
- Medium Speed Data (Sampled at a fixed frequency ~1kHz or less)
  - Thermocouples
  - Slow Speed Pressures
  - Other
- FPGA Data (Sampled based on FPGA setup, engine synchronous and/or asynchronous)

Pre-Processing

- Cylinder alignment of data
- Data scaling to engineering units
- Broad range of signal filtering options
- Cylinder pressure pegging options
- Data windowing option

Pressure Metrics

- Peak Pressure and Location
- Polytropic coefficients of compression and expansion
- Maximum Rate of Pressure Rise and Location
- Pressures at IVO and EVC
- Gross MEP (IMEP), Pumping MEP (PMEP), Net MEP (NMEP)

Heat Release Analysis

- Methods include
  - Single Zone
  - Single Zone Dual Transducer (for pre-chamber engines)
  - Single Zone + Heat Transfer
  - Single Zone Dual Transducer + Heat Transfer
  - Modified Rassweiler and Withrow
  - Pressure Ratio
- Locations of Mass Fraction Burned (5%MFB, 25%MFB, 50%MFB, 75%MFB, 90%MFB, Custom)
- Maximum Heat Release Rate and Location
- Variety of heat transfer correction methods
- In-cylinder temperature estimation

Engine Noise and Knock Analysis

- Raw, time-domain-filtered and angle-domain-filtered noise indication
- Bandpass / Rectify / Integrate / Compare (BRIC) algorithm for knock signal processing
- Offline FFT analysis of knock signals for calibration of BRIC

Installing and Activating DCAT

Please contact Drivven for the latest version of DCAT.

Target System

Recommended Real Time System
Hardware: PXI system, FPGA, S-Series Card
CPU: 2.0 GHz dual core or faster
OS: NI-Real Time 2009
RAM: 1024 MB or more
Hard Drive: 2 GB or more free space

Recommended PC System
Hardware: FPGA, S-Series Card, RTSI Connection
CPU: 2.0 GHz dual core or faster
OS: Windows XP or Vista
RAM: 1024 MB or more
Hard Drive: 2 GB or more free space
Software: LabVIEW 2009, LabVIEW FPGA 2009, DAQmx 9.0

Host System

Recommended System
CPU: 1.6 GHz or faster
OS: Windows XP or Vista
RAM: 1024 MB or more
Hard Drive: 500 MB or more free space
Software: LabVIEW Run-Time 2009, CalVIEW 2.90.xxxx, LabVIEW Full 2009, LabVIEW FPGA 2009, LabVIEW Real-Time 2009, DAQmx 9.0.1
Display: ≥19" wide screen, ≥1440 x 900 (DCAT Only)
≥22" wide screen, ≥1680 x 1050 (Engine Controller and DCAT)

DAQmx 9.0.1	http://joule.ni.com/nidu/cds/view/p/id/1444/lang/en
LabVIEW Run-Time 2009	http://joule.ni.com/nidu/cds/view/p/id/1383/lang/en
CalVIEW 2.86	http://www.drivven.com/visitor_download/Software/CalVIEW2.90.zip

It is assumed that users of DCAT are familiar with internal combustion engine concepts, common practice for acquiring combustion-related data, as well as combustion analysis methods. It is also assumed that the toolkit user is an experienced LabVIEW programmer in the Windows, embedded Real-Time and FPGA environments. However, the DCAT example project is designed so that users with little LabVIEW experience can deploy the project on recommended hardware systems without difficulty.

DCAT is designed to run on a PXI system with the LabVIEW Real-Time operating system. DCAT can also run on a PXI system (or PC) running Windows. However, LabVIEW is limited the 1kHz clock rate (1ms loop rates) in windows.

DCAT requires Drivven's CalVIEW (Calibration toolkit for LabVIEW) to operate. Please reference the CalVIEW User's Manual for Instructions on setting up and activating CalVIEW.

After installing DCAT, it must be activated in order to enable its components. A Computer ID number is generated by the Licensing VI located under the DCAT DAQ pallet. The

Computer ID number must be e-mailed to license@drivven.com along with your name, company name, address and telephone number. Please allow 1-3 business days to validate your account and generate a key. The Computer ID number uniquely identifies the computer being activated based on the MAC address(es) of your computer. An activation key or keys (32 Characters ex. 9DCJ-RN79-QBEE-TPKW-BBAB-GT3Z-RXYC-CFCW) will be generated by Drivven and sent to the customer via reply email. The key(s) will contain only the following characters A B C D E F G H J K L M N P Q R S T W X Y Z 3 4 7 9. The activation key(s) may expire at a specific time if configured to do so.

The activation key(s) can be used in one of three ways. The first method is to wire the activation key(s) directly into the Licensing VI. The second method is to place the activation key(s) in a text file and specify the location of the file in the Licensing VI. The third method is to place the activation key(s) in a text file called “License.dat” in the “c:\ni-rt\system” directory. This allows a real time application to run on multiple systems without modifying the real time code for each system.

The DCAT license VI must be the first DCAT VI to run in a DCAT project. It will validate the DCAT license and unlock the remaining DCAT VIs.

Abbreviations

Async	Asynchronous (sampled at specific time intervals)
ATDC	Crank Angle Degrees After Top Dead Center
BRIC	Bandpass, Rectify, Integrate, and Compare
BTDC	Crank Angle Degrees Before Top Dead Center
CAD	Crank Angle Degrees
cDAQ	Compact DAQ device
CPU	Central Processing Unit
DAQ	Data Acquisition
DCAT	Drivven Combustion Analysis Toolkit
DI	Direct Injection
dV	Differential Volume
EPT	Engine Position Tracking
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
HRR	Heat Release Rate
Hz	Hertz (sec^{-1})
GMEP	Gross Mean Effective Pressure
IMEP	Gross Mean Effective Pressure
kB	Kilobyte
kHz	Kilohertz (10^3 sec^{-1})
MAX	Measurements and Automation eXplorer
MEP	Mean Effective Pressure
MFB	Mass Fraction Burned
MB	Megabyte
MHz	Megahertz (10^6 sec^{-1})
NI	National Instruments
NMEP	Net Mean Effective Pressure
OS	Operating System
PFI	Peripheral Input (DAQ)
PFI	Port Fuel Injector
PMEP	Pumping Mean Effective Pressure
PR	Pressure Ratio
PCI	Peripheral Component Interconnect
PXI	PCI eXtensions for Instrumentation
RIO	Reconfigurable I/O
RT	Real Time
TDC	Top Dead Center
uDCAT	Micro DCAT
Sycn	Synchronous (sampled at specific CAD)
V	Volume
VI	Virtual Instrument

DCAT Hardware Setup

The DCAT is configured to utilize a PXI chassis with a Real Time or Windows based controller, R-Series FPGA card (78XX), and S-Series Cards (612X, etc.). The R-Series card must be inserted into slot number 2 (adjacent to the controller) on the PXI chassis due to the constraints of the PXI bus triggering lines.

The DCAT user must confirm a few configuration settings in the Measurement and Automation eXplorer (MAX) in order to work with DCAT. First, the controller and chassis must be identified (Figure 1). The device names are used to identify the physical channels in the DCAT measurement setup (Figure 2). The device can be renamed by right clicking on the device and selecting the rename option. Last, the PXI Trigger Lines must not be reserved for other functions in MAX (Figure 3). They are automatically reserved as a part of the channel configuration within DCAT VIs.

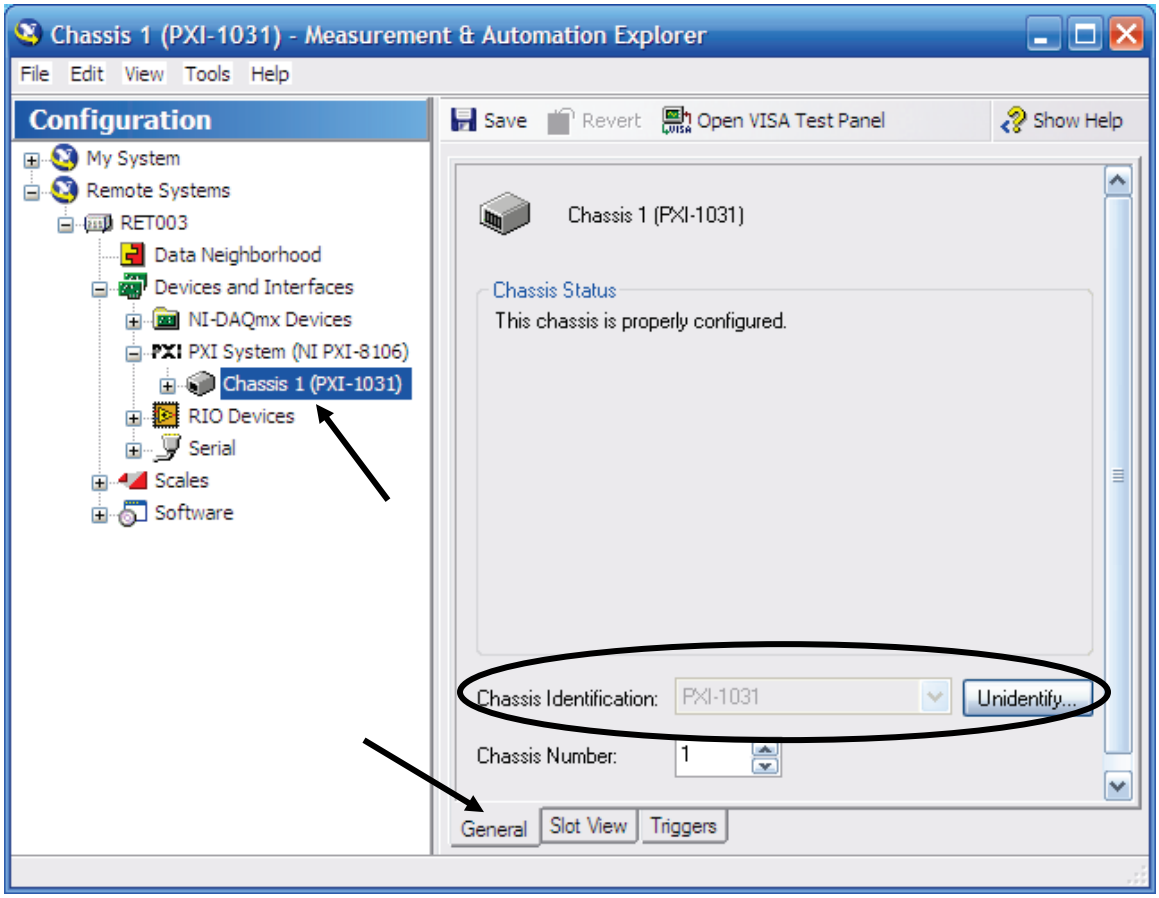


Figure 1: Chassis Setup

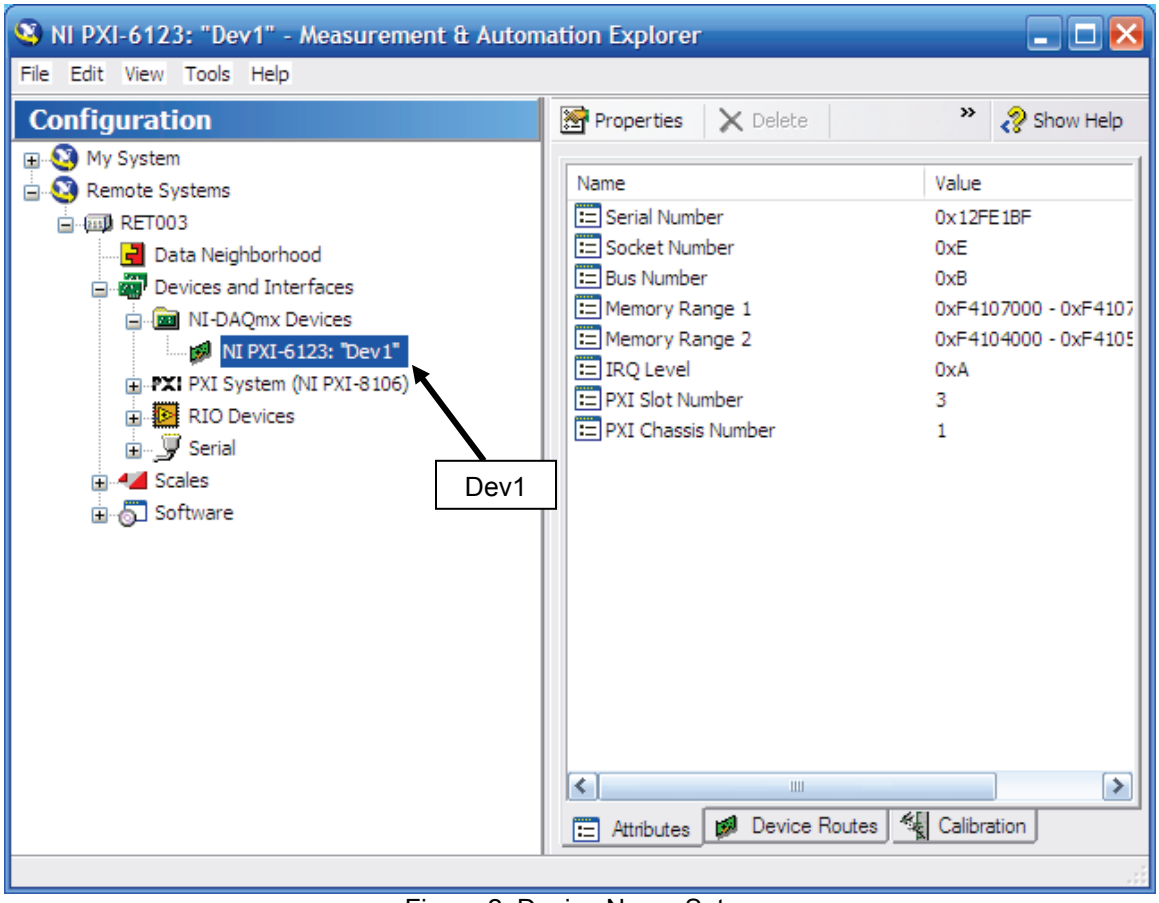


Figure 2: Device Name Setup

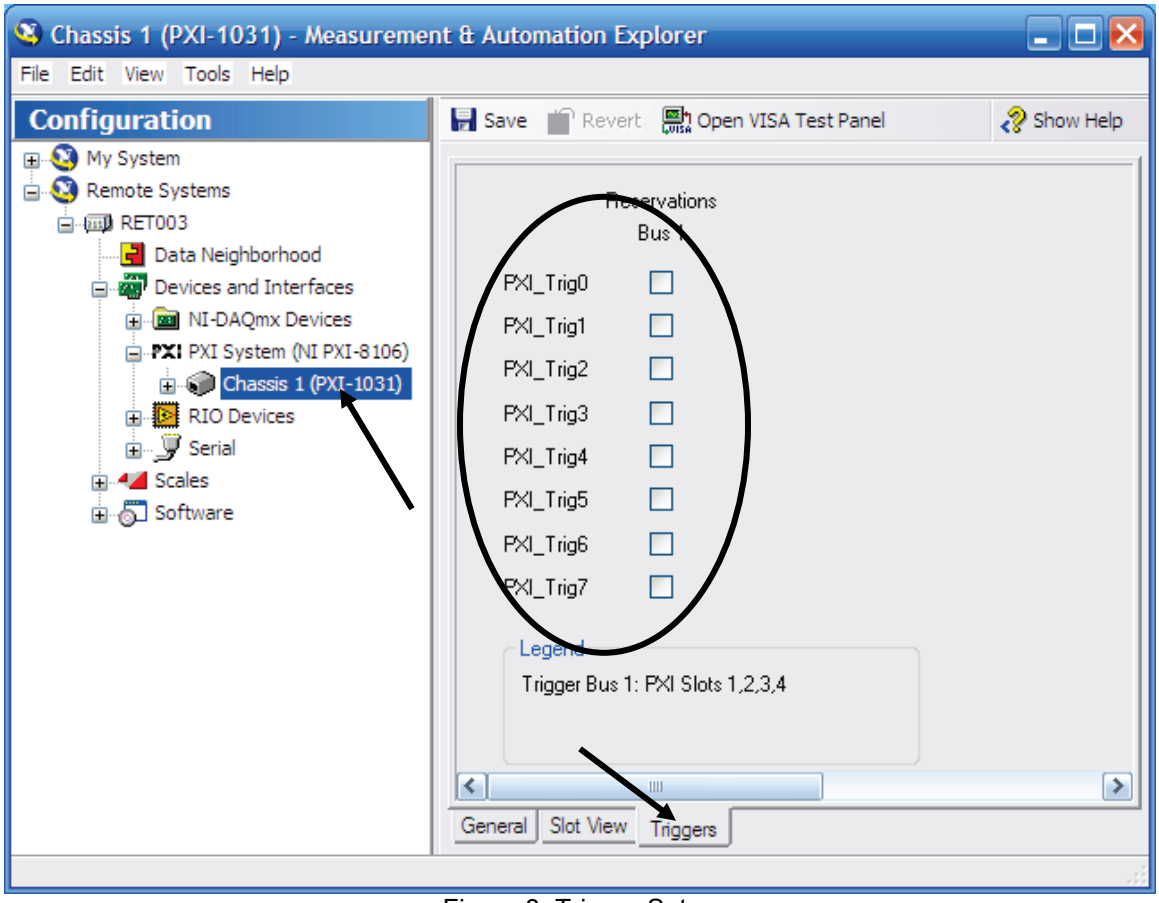


Figure 3: Trigger Setup

Four external wiring jumpers are required for the DCAT system to work correctly. The following pins of the multifunction DAQ accessory must be jumpered on each S-Series card in the system. For the BNC-2090A accessory, 3" of 20AWG wire with ends stripped to 0.25" should be used.

Source	PXI-6122/PXI-6123 Terminals
PFI 03	P0 0
PFI 04	P0 1
PFI 08	P0 2
PFI 09	P0 3
Optional User Input	P0 4
Optional User Input	P0 5
Optional User Input	P0 6
Optional User Input	P0 7

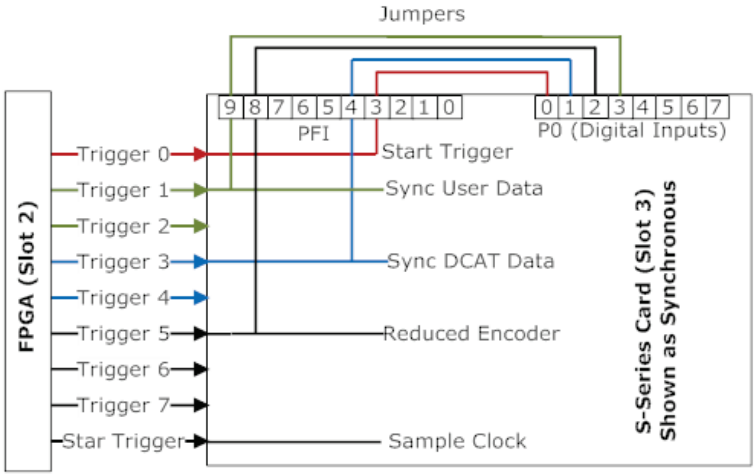


Figure 4: Digital Triggering (Old)

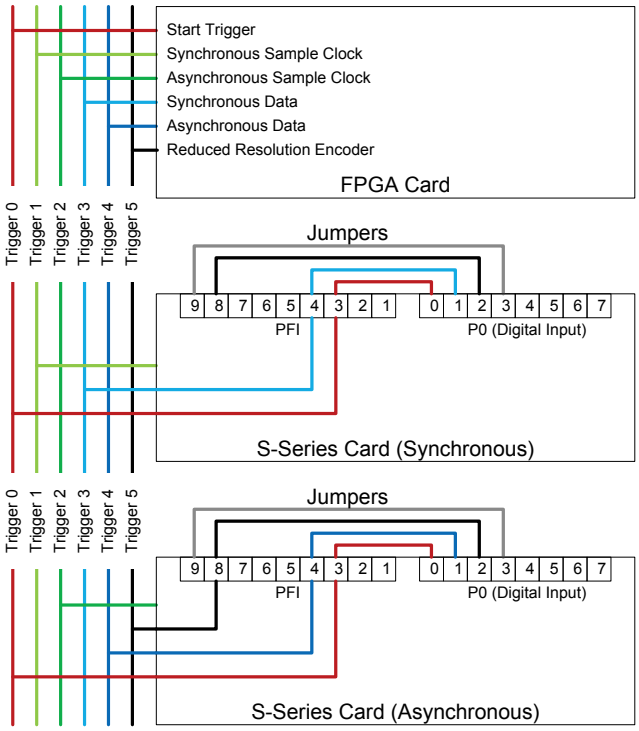


Figure 5: Digital Triggering

For uDCAT hardware setup information, please refer to the uDCAT user's manual.

Encoder Setup and Options

The DCAT toolkit allow for a variety of encoders and triggering options. DAT uses the Engine Position Tracking VI to synchronize the engine sampling. The different EPT VIs require different inputs for different types of patterns. All the EPT VIs provide the required signals to drive the DCAT sampling. This document is concerned only with the encoder pattern. (The encoder patterns are the only EPT VIs that included with DCAT. Please contact Drivven for details about other EPT patterns.) For details on the other patterns, please see the EPT documentation. The encoder EPT requires 2 signals to work properly. The FPGA should be used to condition the signal so that they provide the appropriate signals for the ETP VI. The DCAT toolkit includes several VI that can be used to manipulate the digital signals in the FPGA (Filter, Extend, Offset, etc.)

- Cam: This signal contains a single pulse per cycle (not per revolution). It is used to determine the absolute engine position. The EPT uses the rising edge as the trigger.
- Crank: This signal contains a large number of pulses per revolution. It is used to determine the incremental engine position. The EPT uses the rising edge as the trigger.

The examples included with the toolkit are setup to work with a standard encoder and a cam signal. The crank signal of the encode is in a format that can be used by the EPT. However, the TDC signal is used once per rotation and needs to be once per cycle. Therefore, every other pulse needs to be removed for a 4-stroke engine. There is usually a good place on the cam signal (or any digital signal that has a 720 CAD pattern) to use as a gate for the TDC signal. However, they are not always aligned. To allow a software alignment of the signals, the example uses a Z offset and a cam signal extend. For situations where no cam is required, the cam signal can be set high and the input is not used. All the digital signals have a filter to remove glitches and an invert to allow more flexibility in the setup. Please note that the TDC Offset is from the offset Z signal, not the original.

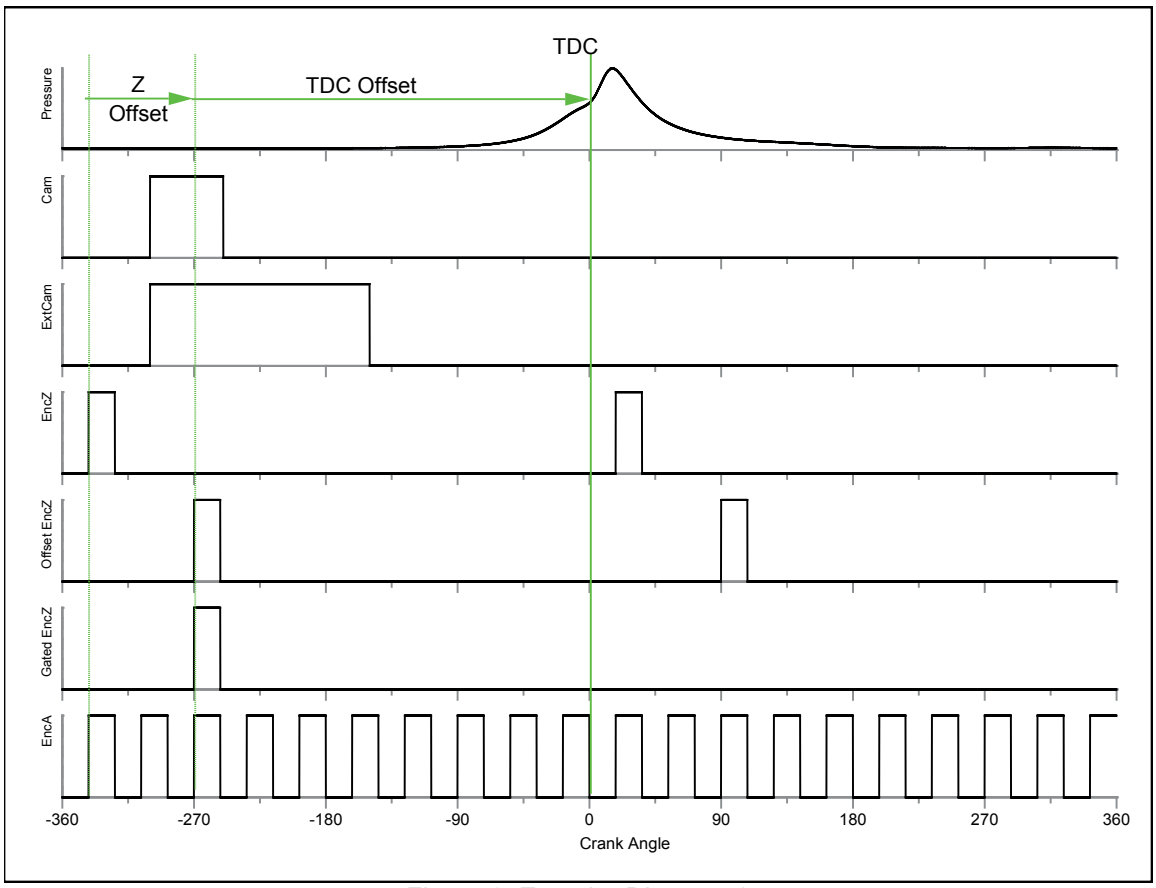


Figure 6: Encoder Diagram 1

Sometimes, the cam signal is not easy to acquire. An alternative may be to use a threshold with the pressure signal (or any analog signal that has a 720 CAD pattern) to get a once per cycle signal. This new signal would replace the cam signal.

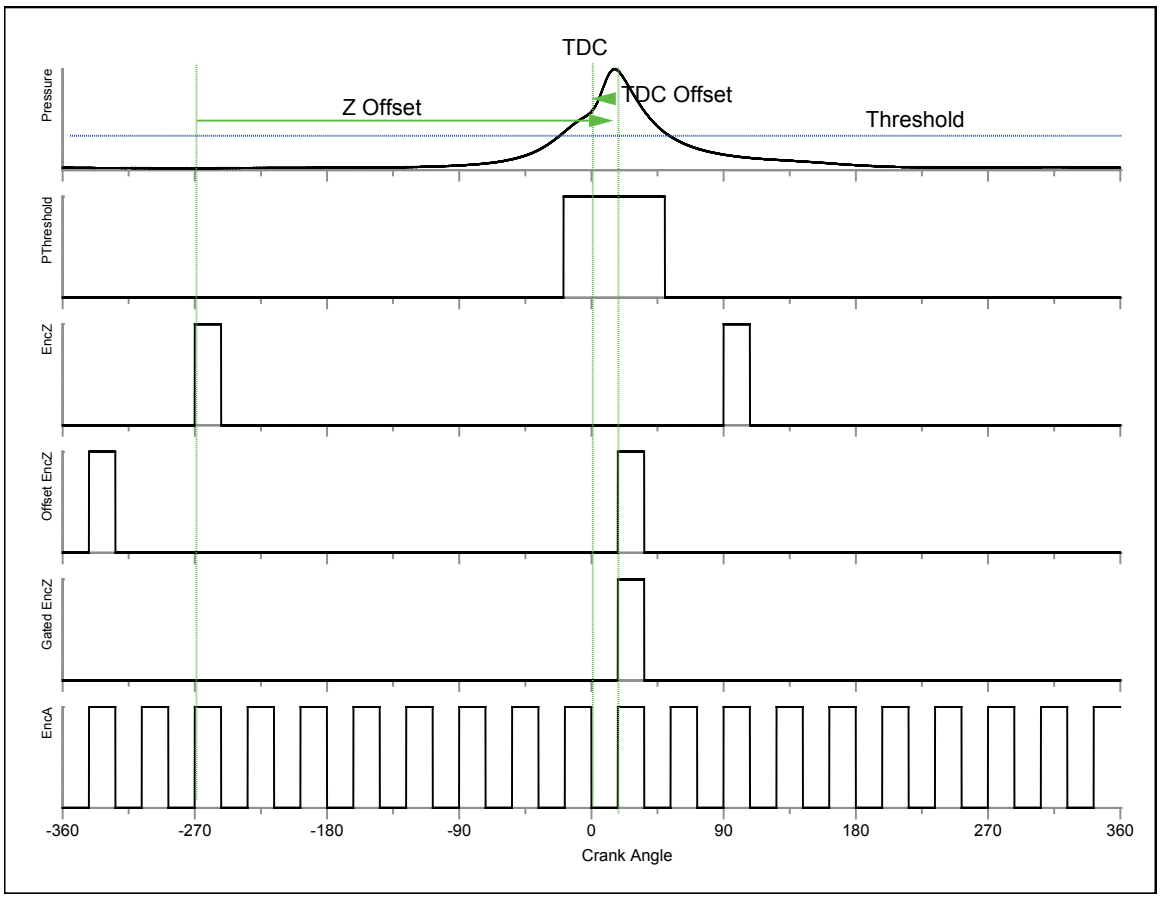


Figure 7: Encoder Diagram 2

The last variation to the FPGA triggering uses a quadrature encoder to increase the sampling resolution. The sampling resolution can also be increased by using the EPT extrapolation. A quadrature encoder can be used to increase the resolution by 2x or 4x. To get a 2x increase in the sampling resolution, the FPGA can be easily set up to watch for a rising or falling edge of the A encoder signal. For a 4x increase in the sampling resolution, the FPGA can be used to watch for a rising or falling edge of the A and B signals.

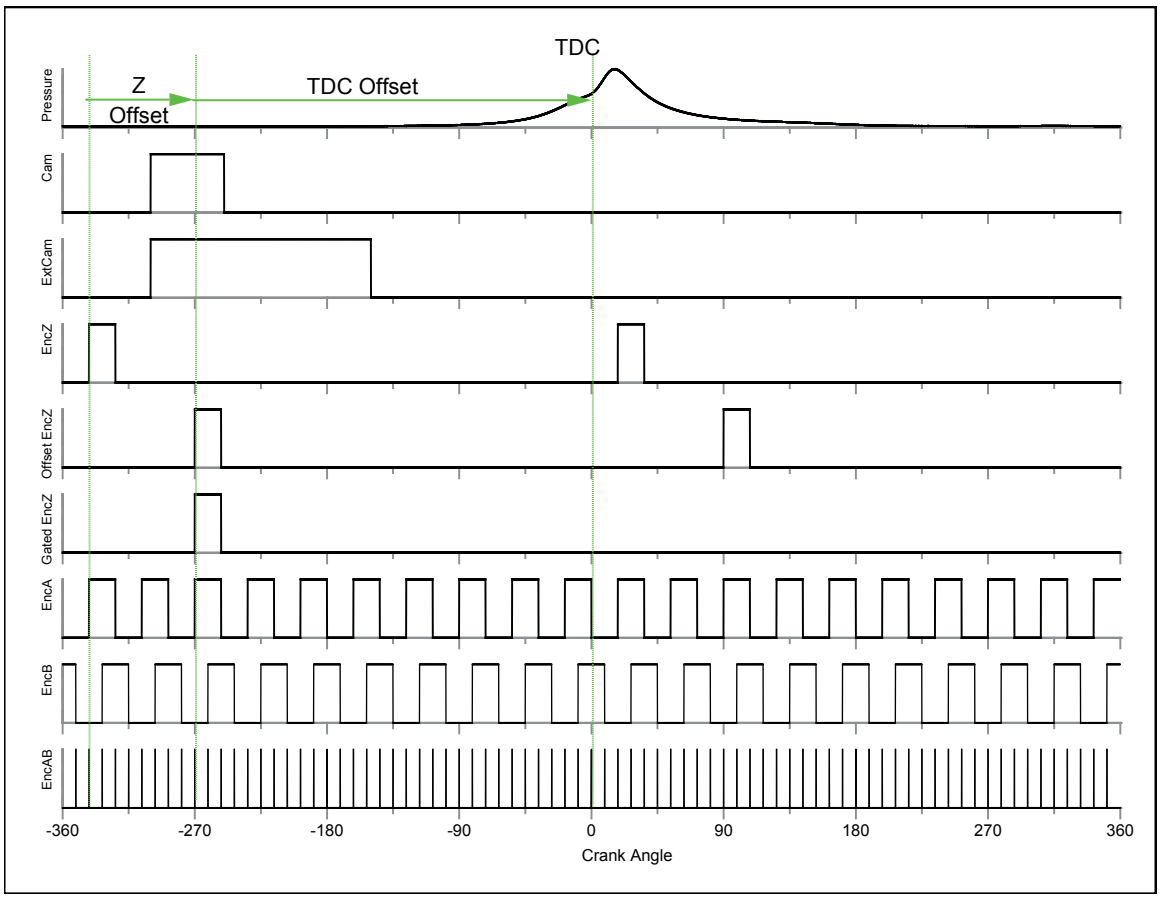


Figure 8: Encoder Diagram 3

The pervious examples are only a few examples of triggering setups. The FPGA allows for a wide variety options to customize the triggering for any application.

TDC Offset Calibration

The TDC offsets are critical for the DCAT to perform calculations accurately. A small shift in the TDC of the pressure waveforms will affect all simple CAD measurements by the amount of the TDC error. These are things like location of peak pressure and location of maximum pressure rise. TDC alignment errors can cause even greater errors on calculated parameters like IMEP and HRR. Some estimates claim that a 0.5 CAD TDC alignment can cause >5% error in IMEP. The TDC alignment is as important as the sensor calibrations.

It is important to understand the relationship between the encoder Z pulse and the TDC. TDC is the encoder A/B pulse directly following the encoder Z pulse. Therefore changing the resolution of the encoder will cause the TDC to shift. The resolution typically changes through one of 2 methods. The first is an encoder that allows the user to select a desired resolution. The second is for the encoder extrapolation to be turned on in the DCAT encoder setup.

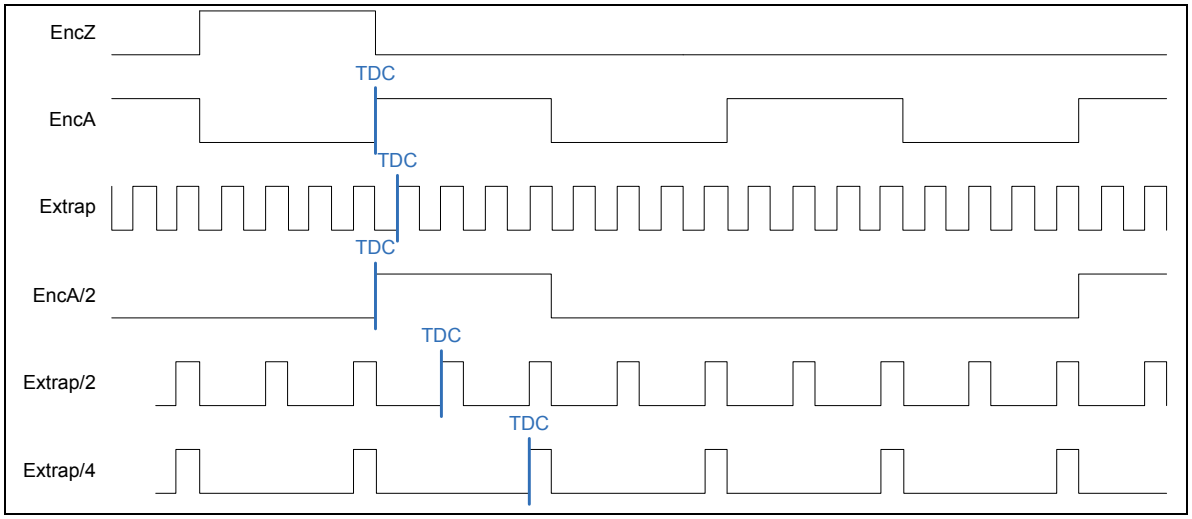


Figure 9: Encoder Z\A TDC relationship

DCAT LabVIEW RT Palette

The DCAT LabVIEW RT palette, found under the Drivven→DrivvenCAT menu, consists of VIs that can be placed in a LabVIEW RT application. Please use the Context Help for detailed information on each of the VIs in the toolkit. The VIs are colored based on their functionality.

Icon Body Color:

DCAT	**Yellow**	(Acquire engine-synchronous and a-synchronous data)
DCAT	**Purple**	File (Saving raw data and summary data to file)
DCAT	**Orange**	Pre-Process (Filtering, etc)
DCAT	**Blue**	Process (Heat Release, etc)
DCAT	**Green**	Statistics (Rolling Statistics, COV, etc)
DCAT	**White**	Control (Custom Controls, CalPoints, etc)
DCAT	**Grey**	Display (Graphs, Charts, Tables, etc)

Icon Title Color:

DCAT	**Yellow**	Custom Control
DCAT	**Red**	Valid for Real Time or Post Processing
DCAT	**White**	Valid for Post Processing Only

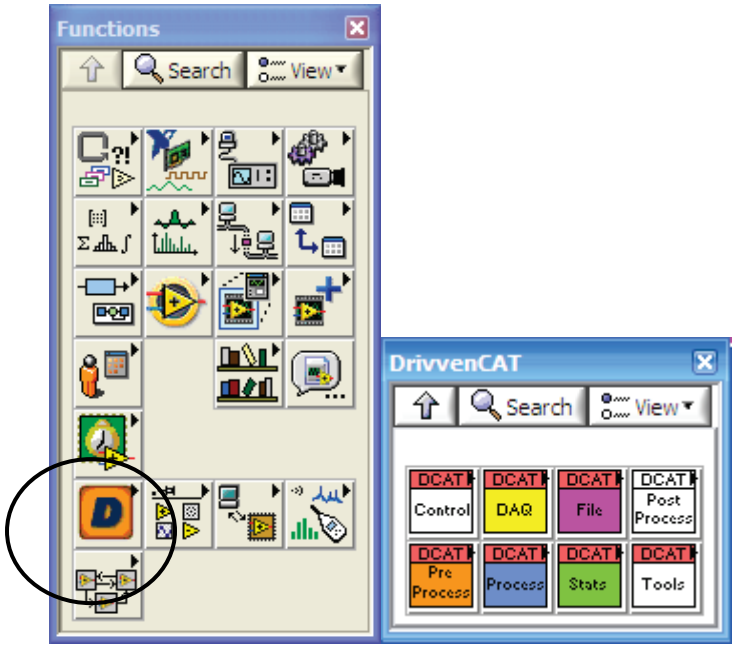


Figure 10: DCAT Pallet Menu

Creating the DCAT Simple Example

The DCAT toolkit was designed to have a simple data flow as shown in Figure 11. After the data is acquired by the DAQ blocks, it is split into two paths. In the downward path, the raw data is streamed to disk using the File palette. In the upper path, the Pre-Process, Process and Statistical palettes are used to filter, analyze and reduce the data, respectively. Each block provides its resulting data to the CalVIEW-based display panel over the network. The statistically reduced data is logged to summary files using the file pallet. The saved raw data file and statistical summary file can be further analyzed off-line using the Post-Process, Pre-Process, Process and Statistical palettes for custom reporting within LabVIEW or Excel. The open architecture of DCAT allows users to pull data out of the Pre-Processing and Processing steps to add their own calculations and displays.

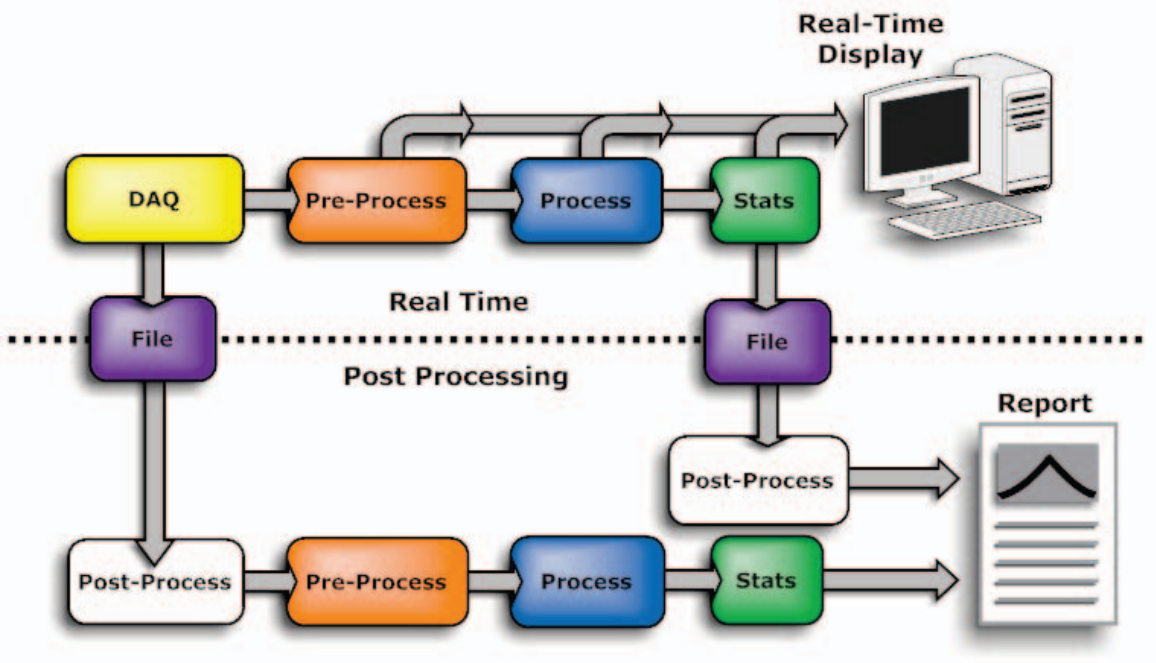


Figure 11: Data Flow

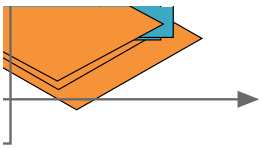


Figure 12: Target Program Flow

We will now walk through the steps of creating a simple DCAT project from scratch…

Please note that some of the VI have large block diagrams and my require opening the files to view their details of the block diagrams.

The example application was designed to:

- Save a summary and raw data files
- Calculate basic pressure metrics (PP, locPP, etc.)
- Use an encoder and cam signal

1. Create a LabVIEW project.
   a. Add a real time target (this example focuses on using the PXI chassis as a real time controller).
   b. Add an FPGA target to the real time target.
   c. Create a user interface VI in the my computer target
   d. Create a target VI on the RT target.
   e. Create an FPGA VI on the FPGA target.
   f. Setup Digital IO for the encoder and the Triggers.

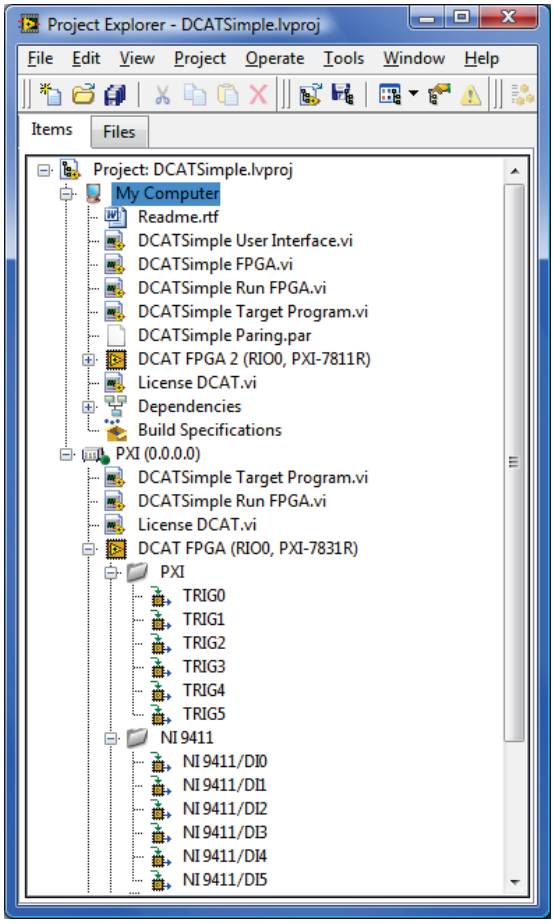


Figure 13: Simple Example Project

2. Next, setup the FPGA VI. This example will use a standard encoder and a cam signal to run the encoder EPT with 2 bits of extrapolation. Many other options are available engine position tracking.
   a. Add a single cycle loop to the FPGA VI. This loop will execute every clock cycle.
   b. Add 3 digital inputs. We need a cam, crank, and TDC signals.
   c. Add digital filters to the inputs. They will remove small noise spikes.
   d. Add a signal extend to the filtered cam signal. This will allow the cam signal to overlap with the TDC pulse. The TDC signal and the cam signal are then ANDed and used as the cam signal for the EPT.
   e. Add the desired EPT and wire all the appropriate controls and indicators. (Multiple EPTs can be compiled into the FPGA so they can be easily changed.)
   f. Add a signal offset to the TDC signal. This allows for alignment of the sampling.
   g. Next, Add the Triggering VI and all the appropriate controls and indicators.
   h. Add the PXI triggers.
   i. Compile the FPGA VI.

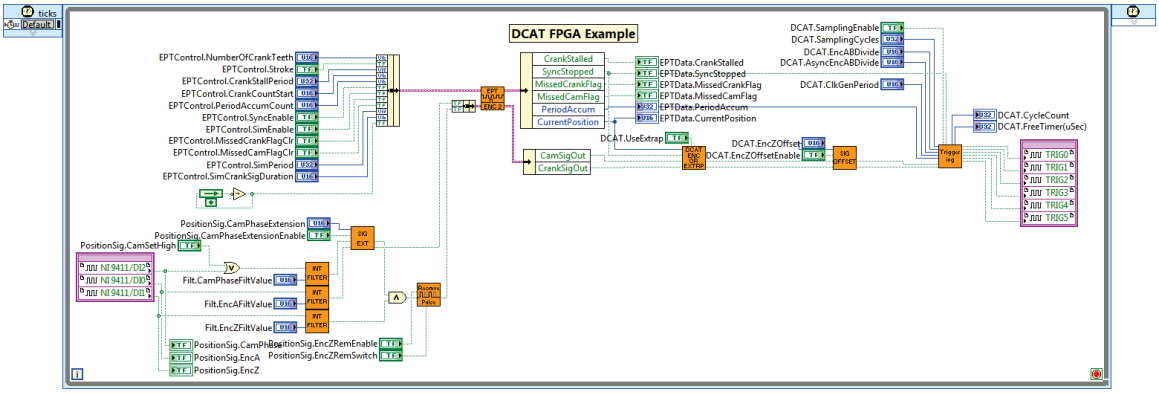


Figure 14: Simple Example FPGA

3. Next, create the basic CalVIEW and DCAT structures in the target application.
   a. Create a case structure. The structure is true if CalVIEW initializes and DCAT is activated. The false case will remain empty.
   b. Add a copy of CalVIEW Master and the license DCAT VI outside the case structure and use their outputs to set the case.
   c. Add a copy of control CalVIEW inside the true case.
   d. Add 2 while loops to the true case of the outer structure. The first loop will run periodically and check a run state CalPoint. The state will write a global variable. The variable is used at various points in the application to stop and start processes. The second while loop holds the DCAT program.
   e. Create a case structure in the second while loop. The case structure should be set by the run state global variable. State 0 is for setup and state 1 is for running.

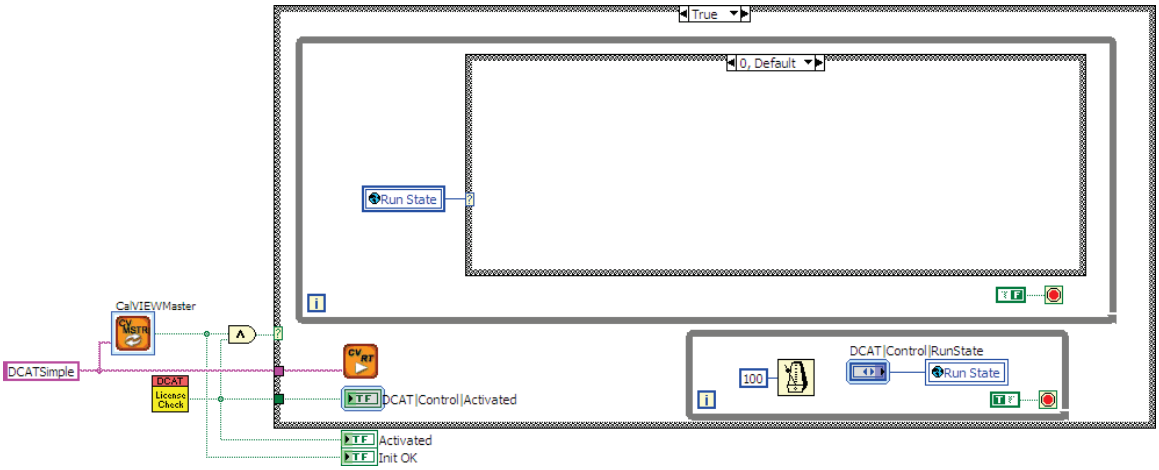


Figure 15: Simple Example Target Program Start

4. Next, add the setup case.
   a. Add the volume array VI. This calculates the volume at every sample of the sampling resolution.
   b. Add the engine setup VI and connect its output to the volume array input.
   c. Add a volume results. It contains CalPoints for each array so they can be displayed and used in the user interface.
   d. Add a timer to keep the VIs from loading the processor.

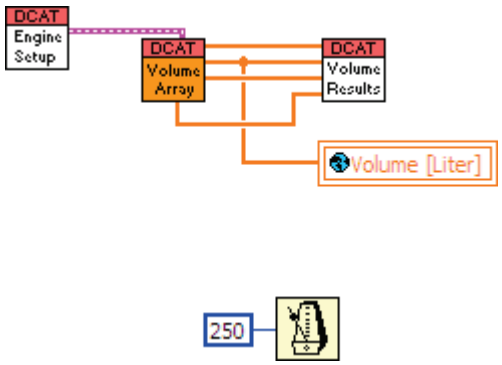


Figure 16: Simple Example Setup

5. Add 3 timed loops to the running case. All of the timed loops should have a resetting timing source.
   a. The first loop should have a loop rate of 3-15 ms and the highest priority. It will be used to collect data.
   b. The second loop should have a loop rate and priority less than the first. It will be used to stream data to the disk.
   c. The last loop should have a loop rate of faster than the cylinder events and the lowest priority. It will be used to calculate real time parameters.

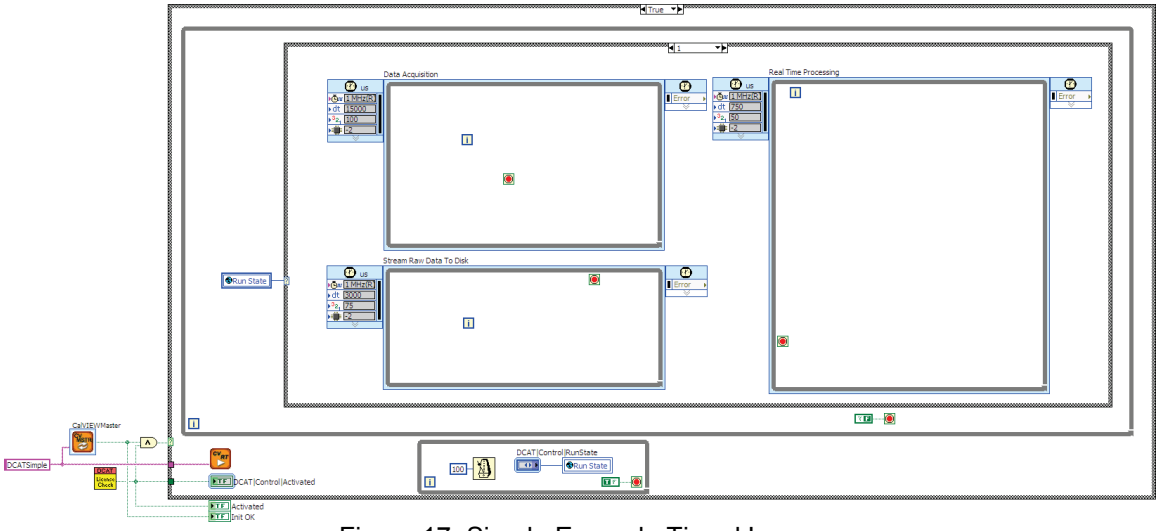


Figure 17: Simple Example Timed Loops

6. Add the DAQ code to the DAQ loop.
   a. Add the acquire data and write all FIFOs VIs.
   b. Add the setup VIs.
   c. Copy and add the Run FPGA VI from the example application. The details of this VI will not be discussed here; please open the VI for more details. The FPGA reference should be updated before the VI will work. The run FPGA VI is responsible for all of the FPGA operations including the following actions:
      i. Initialize and run FPGA
      ii. Reserve PXI Triggers
      iii. Set/Calculate all EPT/DCAT parameters
      iv. EPT synchronization and monitoring
      v. Enabling sampling

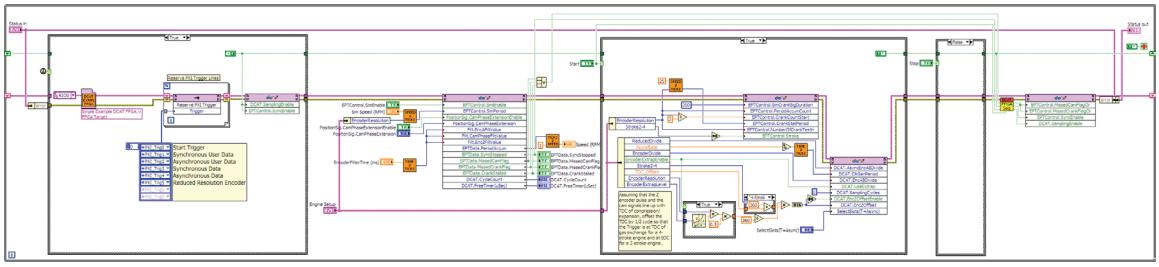


Figure 18: Simple Example Run FPGA

d. Add CalPoints and any desired indicators. The RT queue and ST Queue are used to stop the appropriate loops on an error condition.
e. The loop counter can be used to initialize the VIs as long as the timing source is set to reset.
f. The run state us used to stop the loop.

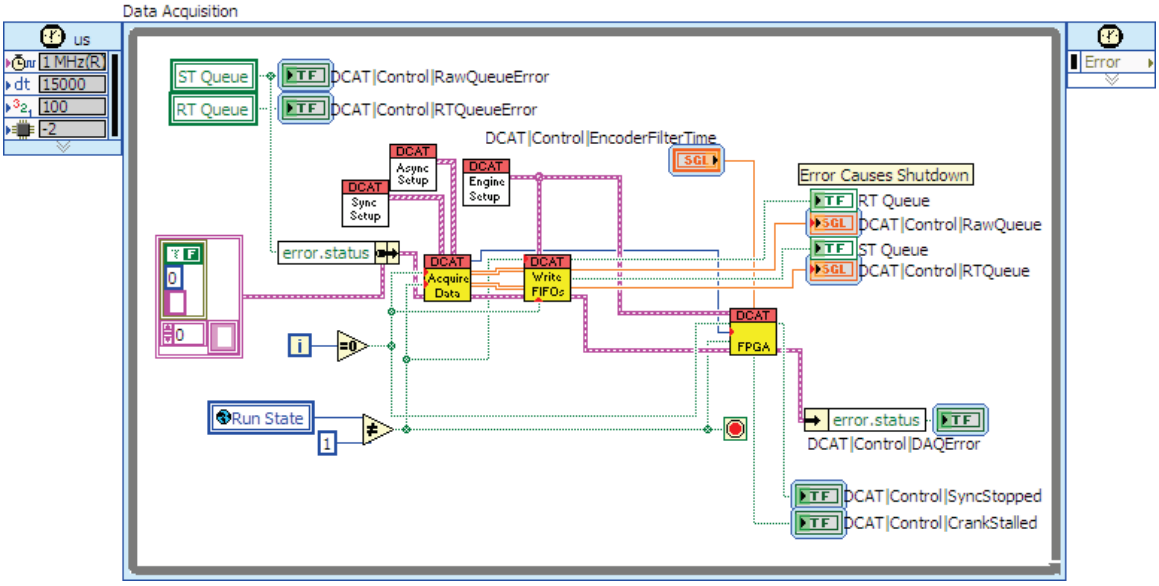


Figure 19: Simple Example DAQ Loop

7. Add the code to save raw data to the streaming raw data loop.
   a. Add the read raw data FIFO VI and the Raw Data to Disk VI.
   b. Wire the outputs of the read raw data FIFO VI to the inputs of the save raw data VI.
   c. Add all of the setup VI that you will use in the program to the loop and combine them into the cluster used by the raw data to disk VI.
   d. Also, add any desired indicators, controls, and CalPoints.
   e. Add an initialization using the loop counter and a stop using the run state and the queue state.

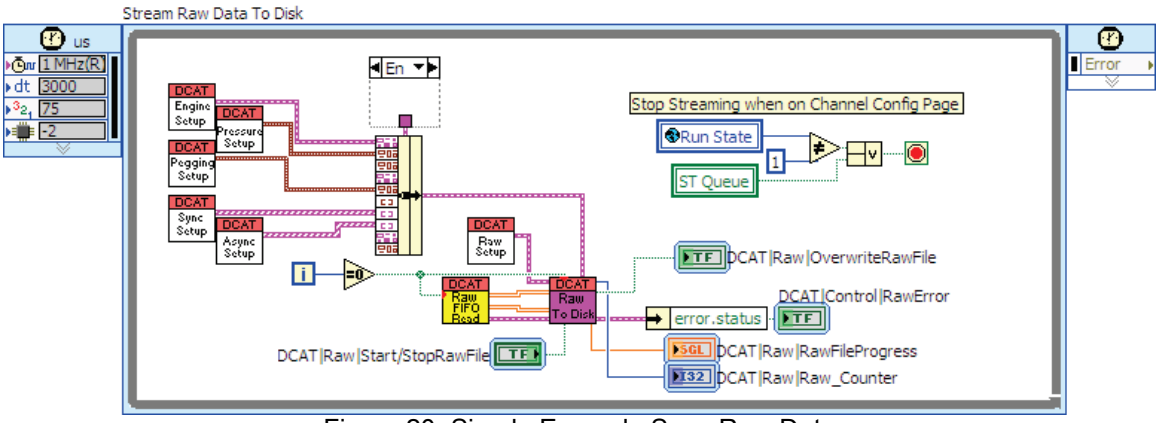


Figure 20: Simple Example Save Raw Data

8. Next, add the code for the real time display and calculations and to save a summary file.
   a. Add the read real time FIFO VI.
   b. Add the Alignment VI. This VI takes the segments of data provided by the FIFO and builds them into useful information. The VI only returns a single cylinder information when available. When no information is available, the VI returns -1 for the cylinder number.
   c. Add a case structure based on the cylinder number. A value of 0, -1, or Default should not execute. A value between 1 and 16 should run the calculations.
   d. Add scaling, filtering and pegging to the valid cylinder number case.
   e. Add pressure metrics VI, the pressure metrics statistics VI, and the pressure metrics results VI. More sets of calculations may be added.
   f. Add the array results and the motoring results.
   g. Add all of the setup VI required by the different calculation
   h. Add the save summary file VI and create a cluster of the setup clusters.
   i. Add all desired indicators, controls, and CalPoints.
   j. Use the loop counter as the initialization.
   k. Use the run state and queue error as the stop conditions.

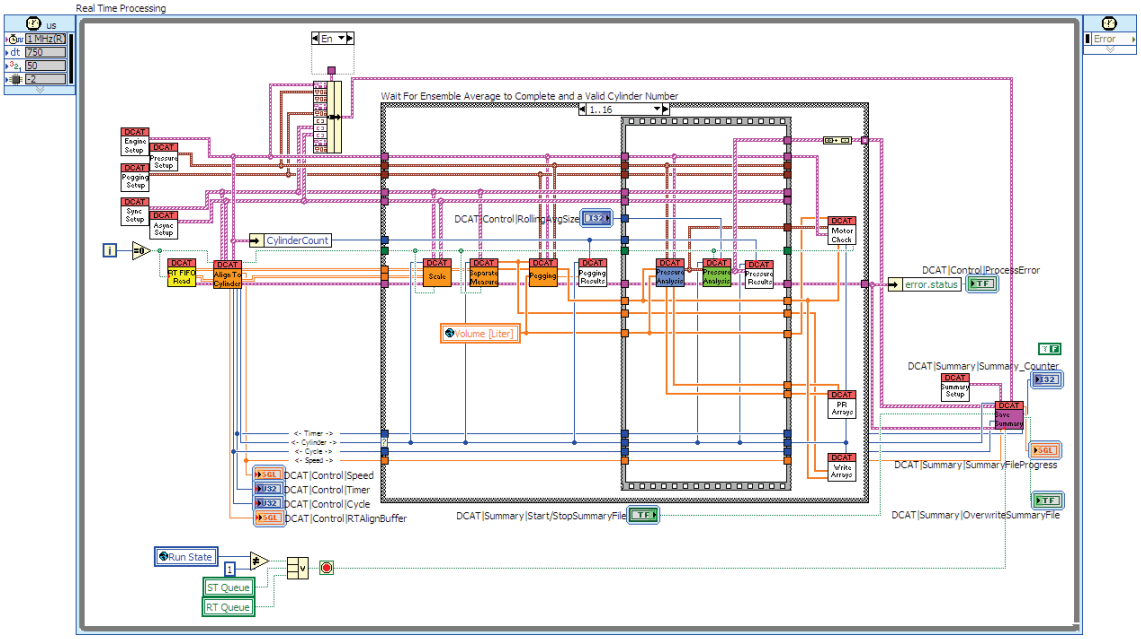


Figure 21: Simple Example Real Time Calculations

9. A user interface is now needed. The user interface runs on the host machine using CalVIEW. All CalPoints on the target need a matching indicator or control on the host panel. In this example, the user interface will contain a tab control to define the state of the target program. The first two tabs (measurement setup and engine setup) will cause the target application to enter the setup state and the rest of the tabs will cause the target program to enter the running state. For organization, the tab control will control a case structure on the block diagram containing any calculations and controls needed on the corresponding tab. Some tabs will not require any logic in their case but others may for formatting table, graphs and charts. A second loop will be used to show trends. Please refer to the CalVIEW documentation for details about creating a user interface.

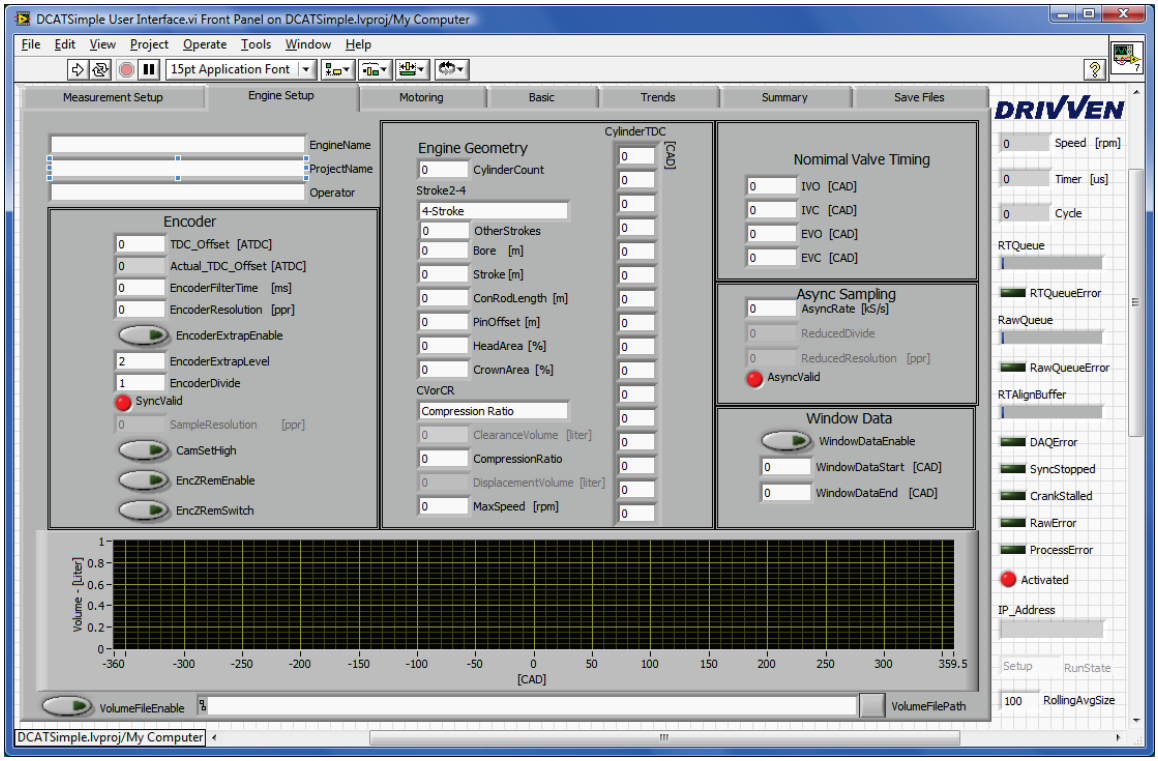


Figure 22: Simple Example User Interface

10. Create a pairing file.
    a. Open CalVIEW if it is not already open.
    b. Select the proper target and host VI.
    c. Pair the target and host items
    d. Refer to the CalVIEW user's manual for more details.

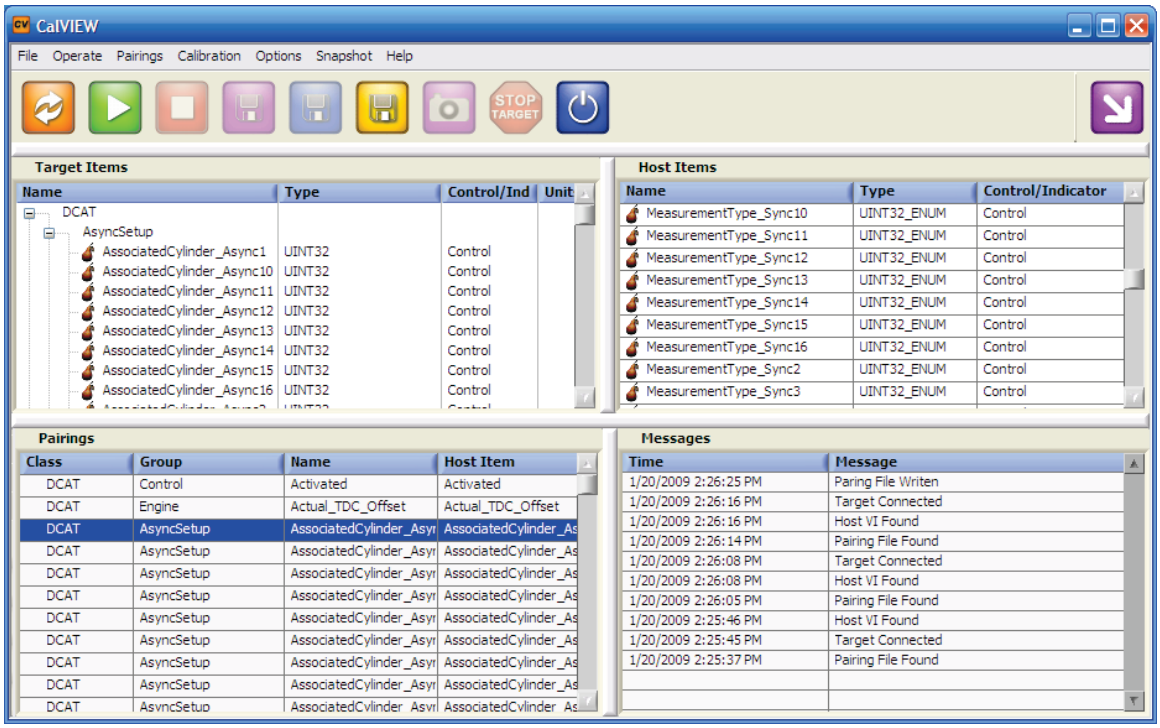


Figure 23: Simple Example Pairing

11. Run the host application using CalVIEW.

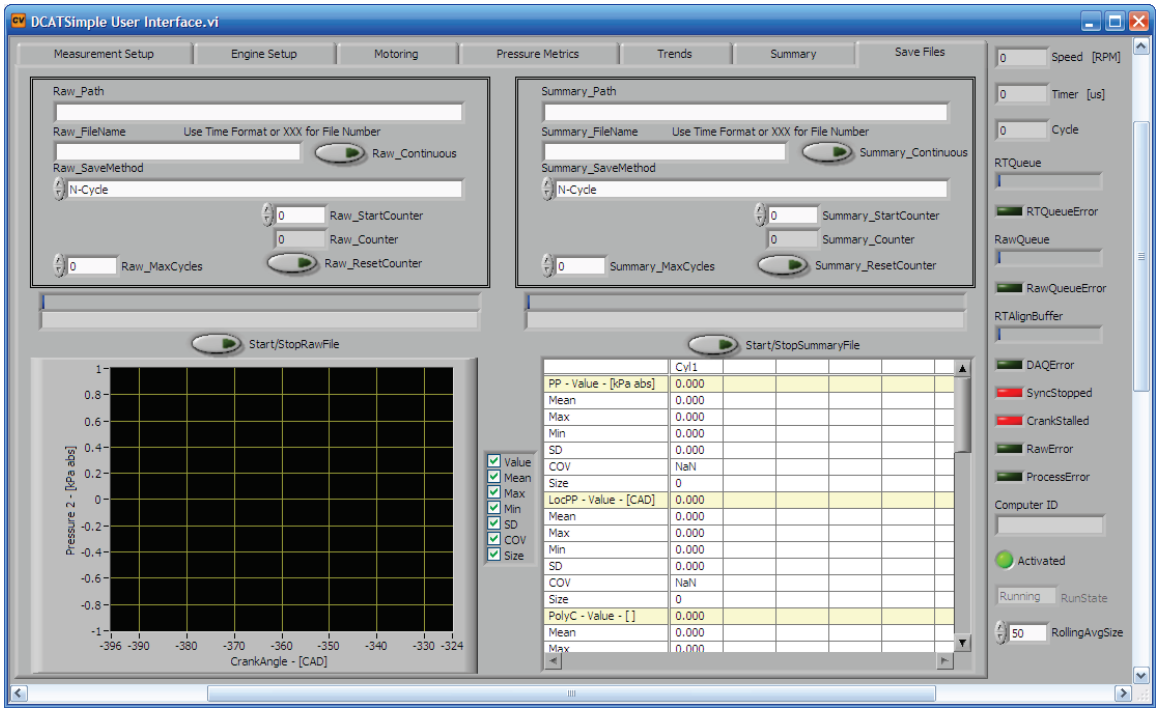


Figure 24: Simple Example User Interface

Reviewing the DCAT Advanced Example

The DCAT Advanced example uses the same general data flow with the addition of an optional third path. The third path in Figure 25 is used for next cycle feedback control. In the next cycle control path, the DAQ alignment is stopped early to allow the calculations to finish before the next cycle begins and only a few select calculations are preformed. The example application also contains an estimate of the finish of the calculations to ensure that the calculations are finished at a reasonable crank angle.

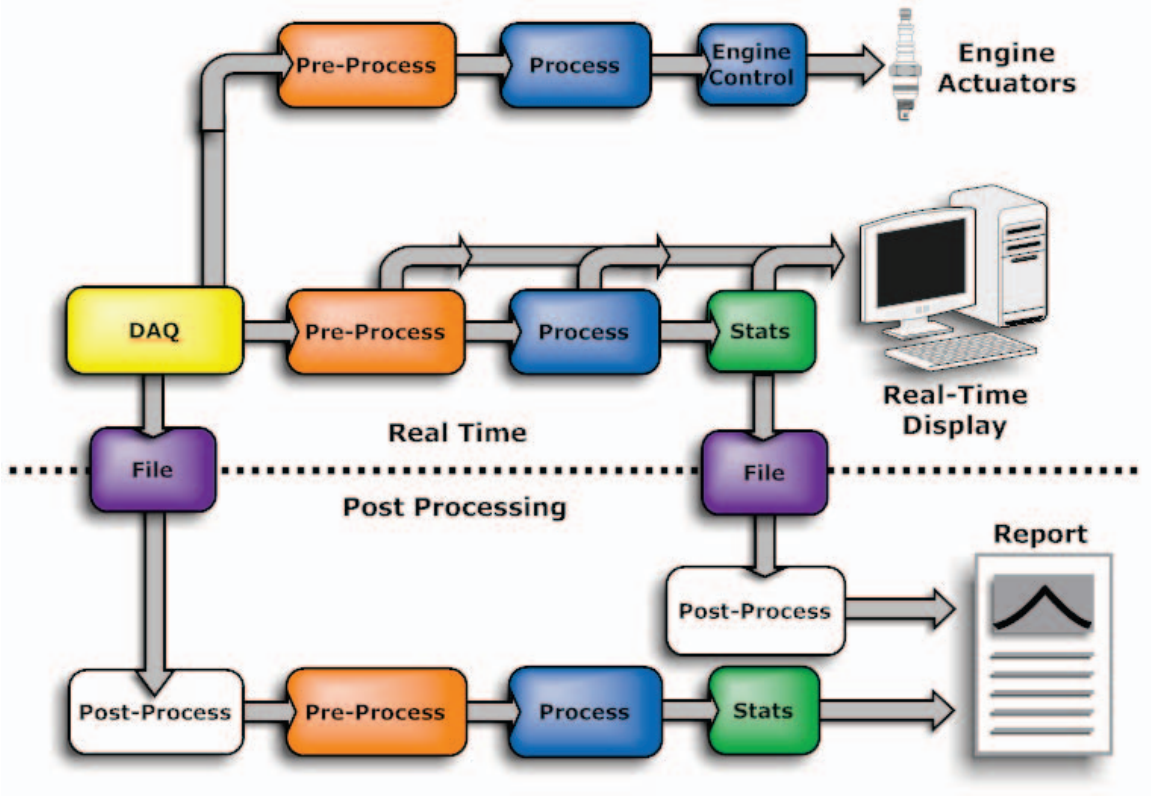


Figure 25: Data Flow with Next Cycle Control

The advanced DCAT example demonstrates the use of the majority of VIs within the toolkit. The example can be used with little or no modification when applied to the recommended PXI hardware as a powerful combustion analysis application. The example can also be modified to fit a specific requirement. This section provides an overview of the advanced example application. When creating a custom advanced DCAT application, it is recommended to use this application as a template. For most applications, the processing loop will be the focus of the modifications. Please refer to the appendix at the end of this manual for detailed information on the individual SubVIs. Due to the large size of the block diagram, it is recommended that the actual VIs be opened in order to see details. In the following figures, labels are placed on the block diagram screen shots corresponding to comments below.

The DCATAdvanced example may be run on the recommended hardware platform in a few short steps:

1. Open the example project file “Advanced Example DCAT Project.lvproj”
2. Configure the IP address of the PXI RT Target within the target properties dialog
3. Compile the FPGA
4. Open the top level PXI RT target VI “Advanced Example Target Program.vi”
5. Run CalVIEW Master on the target program.
6. Connect to the RT target and Run the top level VI
7. Open CalVIEW and connect to the target, load the host VI “Advanced Example User Interface.vi” and load the pairing file “Advanced Example Pairing.vi”
8. Within CalVIEW, start the host VI to begin combustion analysis setup and operation

Reviewing the DCATAdvanced Target VI

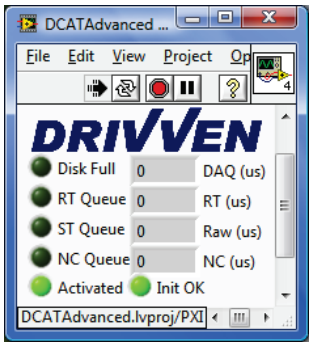


Figure 26: Target Application Front Panel

We will now explore the top level RT VI. The top level RT front panel contains only a few indicators for status information for the user. However, this is not the intended user interface for the DCAT application. The actual user interface for manipulating combustion analysis tasks is executed on a windows host PC, supervised by CalVIEW, and will be discussed later in this document. The Boolean indicators communicate the status of the queues and disk among the real time loops. The four numeric indicators show the execution times of each loop. Figure 27 below shows the entire block diagram for the top level RT VI. Further diagrams below will show close-ups of the diagram portions being discussed.

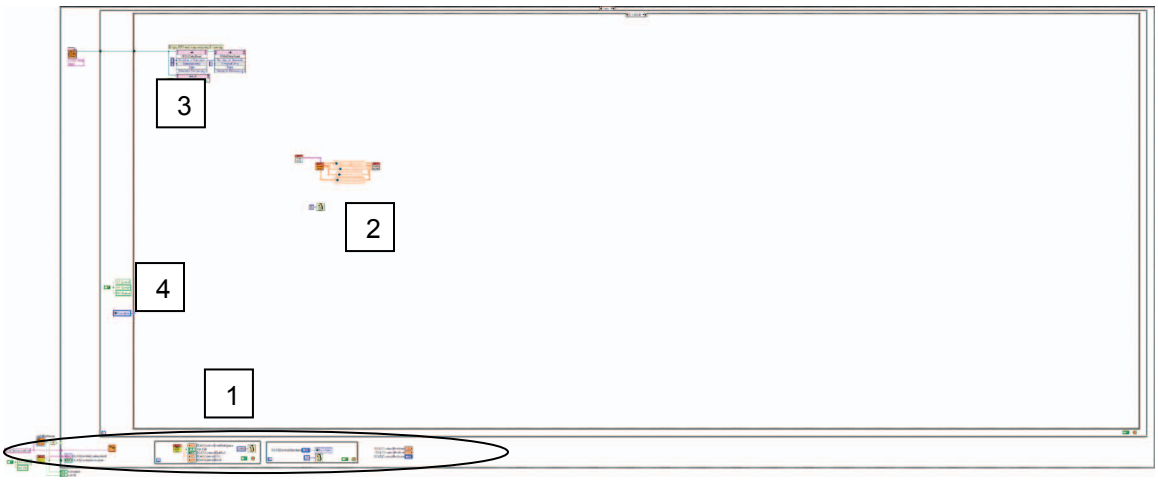


Figure 27: Top level RT block diagram (RunState = Setup = 0)

1. This section of the code (including the outer case structure and the two small while loops) initializes CalVIEW and monitors the memory and CPU usage of the real time controller (lower while loops). Please see the CalVIEW User's Manual for detailed instructions on using CalVIEW.

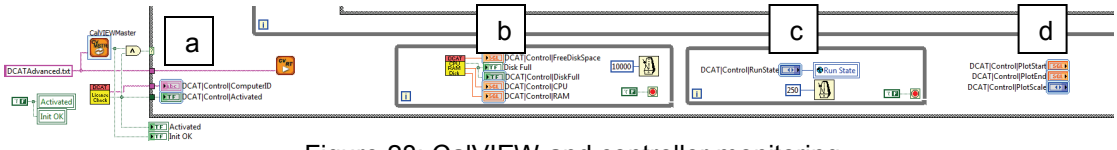


Figure 28: CalVIEW and controller monitoring

a. CalVIEW control and CalVIEW Master. The CalVIEW master should control the outermost case structure. The Control CalVIEW VI should be outside all loops. Please see the CalVIEW manual for details.
b. Real time controller status. Gets the current CPU, RAM and HDD use. On windows systems, only the HDD information is available.
c. Checks the DCAT state. The two states are Running and Setup.
d. CalPoints to hold display configuration information.

2. This section of code runs periodically when the program is in setup mode. This section of code calculates all of the parameters that can be set during setup. It currently only contains geometry calculations.

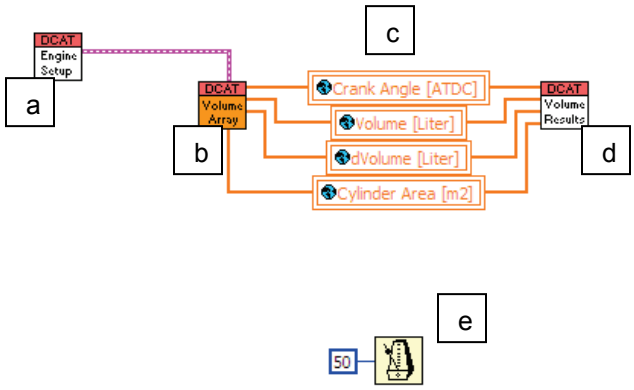


Figure 29: Setup Calculations

a. Engine setup cluster is created using CalPoints.
b. Crank Angle, Cylinder Volume, Differential Cylinder Volume, and Cylinder Area arrays are created using the engine geometry. The size of the arrays corresponds to the sampling resolution.
c. The arrays are written to global variables to be used in other parts of the program.
d. The arrays are written to CalPoints to be used and displayed on the user interface.

3. FPGA Initialization and clear FIFOs: This section of code initializes and runs the FPGA. The FPGA reference is used to access and control the FPGA parameters throughout the program. The FPGA FIFOs are cleared during the setup mode so that they are ready to run when switched to the run mode.

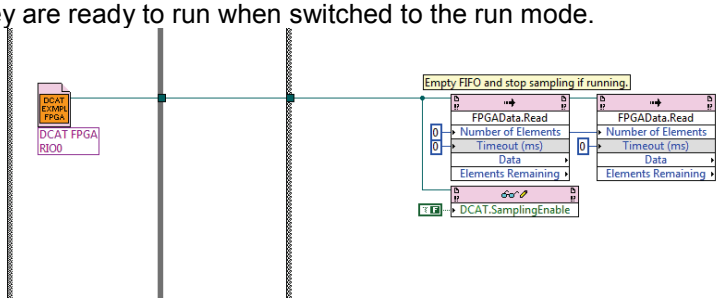


Figure 30: FPGA Initialize

4. Running State: This section of code initializes the Queue states and sets the running state.

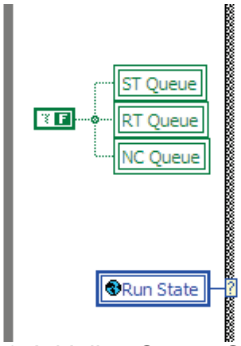


Figure 31: Initialize Queue States

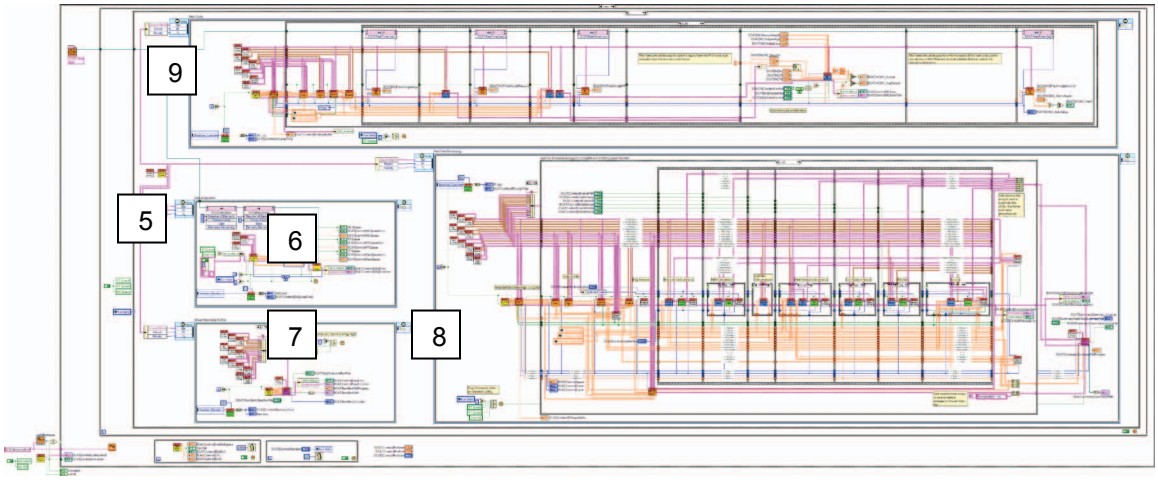


Figure 32: Top level RT block diagram (RunState = Running = 1)

5. Timed Loop Setup: This VI uses some of the engine setup information and calculates the desired loop time for each of the four timed loops.

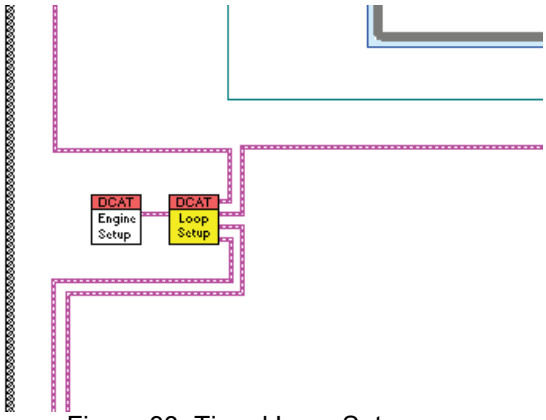


Figure 33: Timed Loop Setup

6. DAQ Loop: This loop is the highest priority loop of a DCAT application because all other loops and tasks depend on the timely data it acquires.

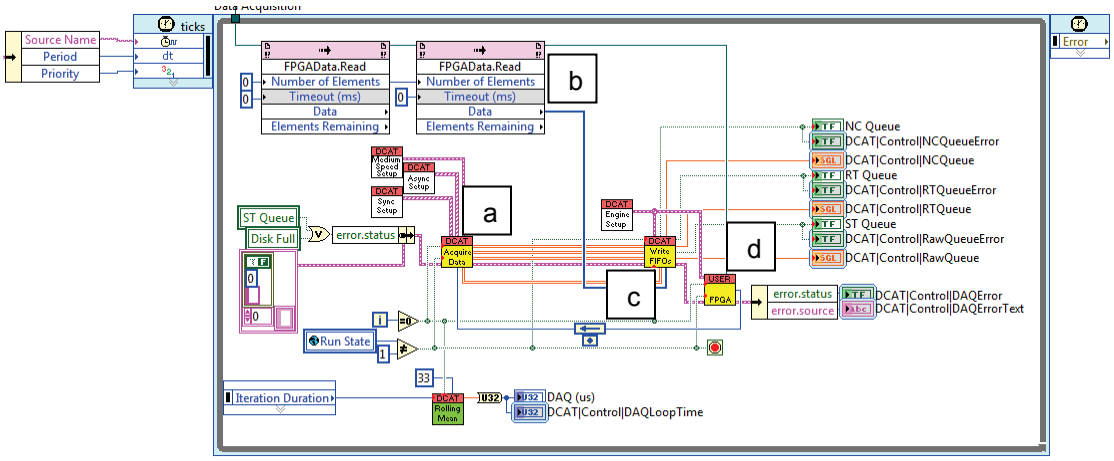


Figure 34: Data Acquisition Loop

a. Acquire Data: Configures the DAQmx tasks and hardware timing based on the user defined settings. The VI returns the entire available DAQmx buffers each time the VI runs.
b. Read FPGA Data FIFO: Reads the all the elements in the FPGA Data FIFO and passes them to the quiues.
c. Write All Queues: The data is written to a set of queues to be read in other parts of the application. The status of the queues is also checked to verify that they have not completely filled or caused an error.
d. FPGA Control: All FPGA interface is handled through this VI. The FPGA is used to generate the engine-synchronous and time-synchronous clocks to the DAQ cards.

7. Streaming Loop: The Streaming timed-loop is set to a lower priority than the DAQ loop but higher than the processing loop.

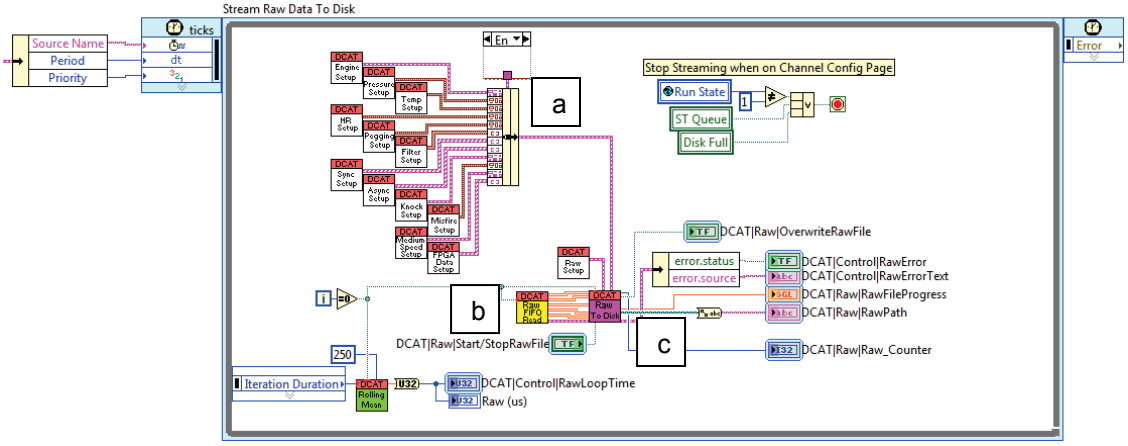


Figure 35: Streaming to Disk Loop

a. Setup parameters: All of the setup parameters are bundled into a cluster that is saved at the beginning of a file.
b. Read the queue.
c. Stream available data to the disk.

8. Processing Loop: This loop is responsible for the real time processing and display of the data. It also saves the summary file.

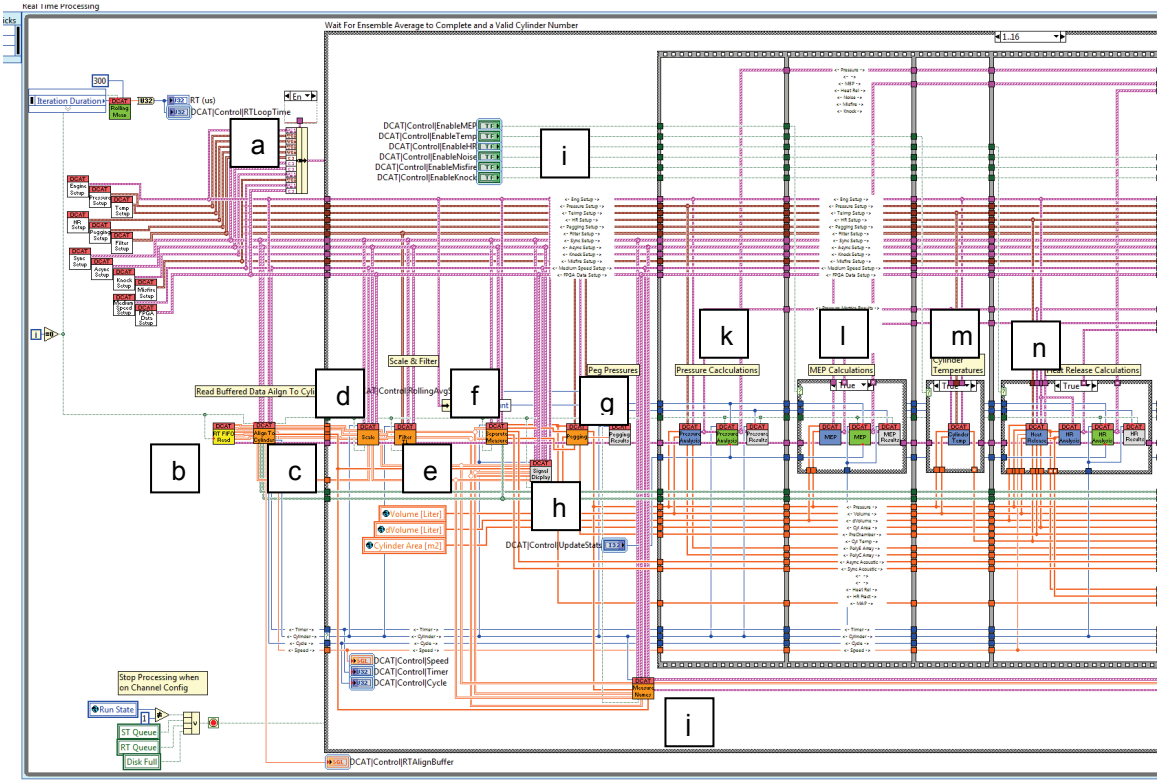


Figure 36: Real Time Processing Loop (a)

a. Combine setup properties to be saved with each file.
b. Read queue.
c. Align Data to cylinder: This VI buffers the data and aligns the data based on TDC offsets for each cylinder. The VI provides a cylinder number of -1 when it doesn't have a full cycle of data and provides a valid cylinder number when there is a full cycle of data. Only one cylinder of data can be presented each loop.
d. The data is scaled.
e. The data is filtered.
f. Separate the cylinder pressure, pre-chamber pressure, manifold air pressure, and acoustic data from the arrays.
g. Cylinder pressure pegging.
h. Write all data arrays to CalPoints for display.
i. Option to disable different sets of calculations.
j. Overwrites pressure with pegged pressure and gets a unique name.
k. Pressure Metrics Calculations.
l. Mean Effective Pressure Calculations.
m. Cylinder Temperature Calculations.
n. Heat Release Calculations.

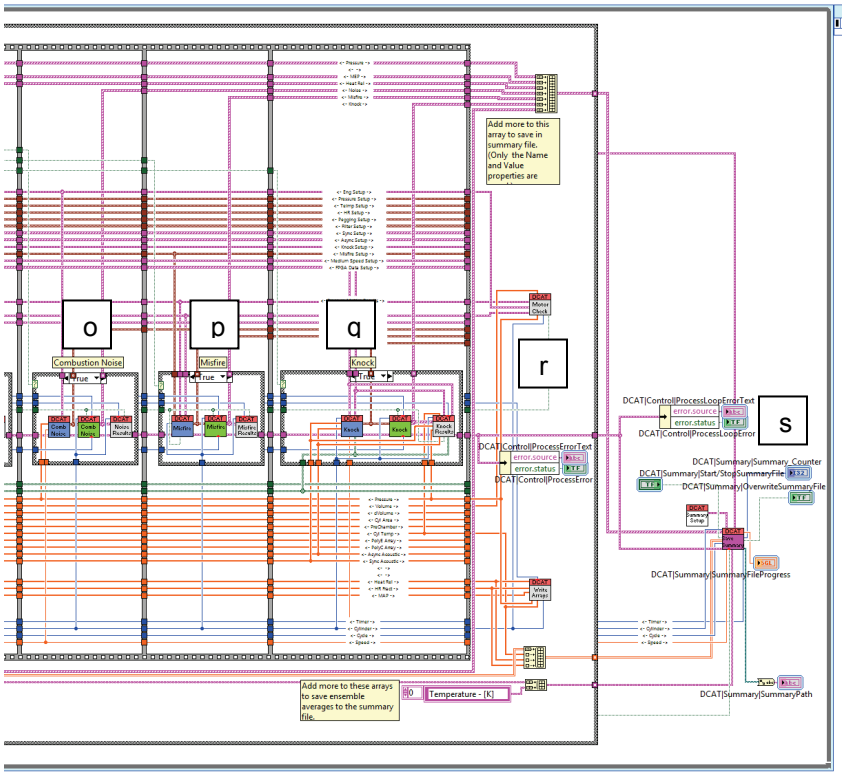


Figure 37: Real Time Processing Loop (b)

o. Combustion noise calculations.
p. Misfire calculations.
q. Knock calculations.
r. Cycle data arrays for the user interface.
s. Summary file saving.

9. Next Cycle Control
   a. Read queue and align data.
   b. Preprocessing data.
   c. Calculate current crank angle estimate.
   d. Calculate the first parameter
   e. Calculate the second parameter
   f. Read inputs, control strategy, and write outputs.
   g. Calculate the final crank angel estimate.

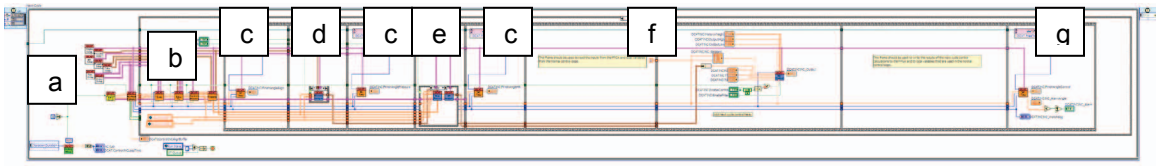


Figure 38: Nest cycle Loop

Reviewing the DCATAdvanced FPGA VI

The FPGA application is used to control the triggering signals to the DAQ cards. The FPGA can be configured to the handle all common sampling setups. DCAT was designed with the possibility of implementing engine control within the same system. Therefore, the EPT VI used for DCAT may also be used to supervise other Drivven I/O modules for fuel and spark control. Please refer to the documentation for those I/O products for further information.

In general, the example DCAT FPGA VI uses an encoder A signal, an encoder Z signal, and a cam phase signal to generate triggers and sample clocks to the DAQ cards over the PXI bus. However, FPGA technology used by DCAT enables many additional triggering and synchronization capabilities. Some variations include the following:

- Using the rising and falling edges of encoder A and B signals to increase the resolution by up to 4X.
- Use Drivven's Engine Position Tracking (EPT) VIs to track the angular position of encoders or production engine flywheel patterns to provide extrapolated high resolution sample clocks to the DAQ cards.
- Employ sophisticated custom filter of position signals.
- Extend, delay or synchronously offset position based signals for precise encoder alignment without the need to mechanically adjust the physical signal source.
- Sample digital control parameters or signals, originating in the FPGA, in sync with the external analog signals. For example, the commanded fuel pulse duration can be encoded and sampled with each engine cycle of data so that it will be known while processing the combustion data offline.

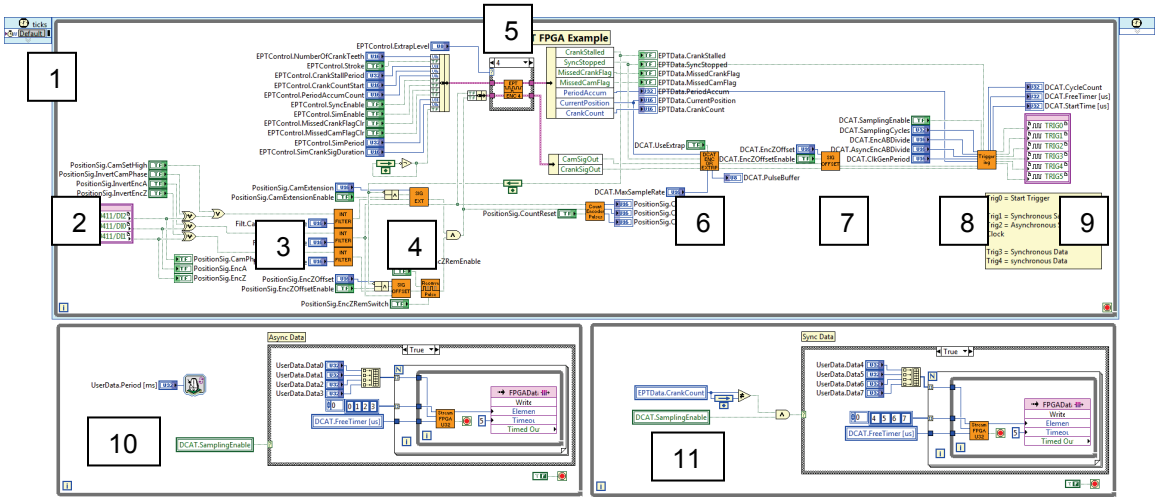


Figure 39: Top level FPGA block diagram

1. Single Cycle Loop: Entire loop executes at 40 MHz.
2. Position Signals: Read encoder and cam signals from NI-9411 differential digital input module. Any digital input source will work.
3. Filter position signals.
4. Cam signal Extension: This VI will take a single narrow pulse from the camshaft and extend it in order to gate every other encoder Z pulse.
5. Engine Position Tracking (EPT): Please see the EPT documentation for details on these SubVIs. The EPT software can use several levels of extrapolation as designated by the number on the icon. The number represents the binary bits of extrapolation used.
6. Extrapolation Option: This VI allows the user to select whether the raw encoder signals or higher resolution extrapolated position signals are used for clocking the engine-synchronous DAQ cards. If the bits of extrapolation is X then the raw encoder pulses are increased by (2^X)/2.
7. TDC Offset: This VI offsets the TDC position signal so that a mechanical adjustment is not required. When using extrapolation, the offset resolution is determined by the EPT extrapolation amount.
8. Triggering: This VI controls all the generation of all the synchronous and asynchronous triggering signals. It is responsible for encoder resolution dividing and encoding data on the trigger lines.
9. Clock and Trigger I/O Nodes: Assign sampling and trigger signals to the appropriate DAQ card slots based on the DAQmx setup. These signals are carried over the PXI bus.
10. Asynchronous FPGA Data FIFO: Write data to the FGPA Data FIFO at the specified sample rate when the DCAT sampling is enabled.
11. Synchronous FPGA Data FIFO: Write data to the FGPA Data FIFO at specified crank angles when the DCAT sampling is enabled.

Reviewing the DCATAdvanced User Interface VI

The advanced DCAT user interface was created using the process documented in the CalVIEW user's manual. It contains a tab control which organizes the controls and indicators according to DCAT functionality. The controls and indicators correspond to CalPoints within the RT application and are communicated between the RT target and the host by CalVIEW.

The DCAT toolkit contains several user interface VIs which are useful for the setup and control of strip charts, plots, and tables within the user interface. These VIs have a grey body and are found in the tools DCAT\Tools pallet. For details, please see the help for each of the VIs.

The example user interface contains several aspects that should be noted.

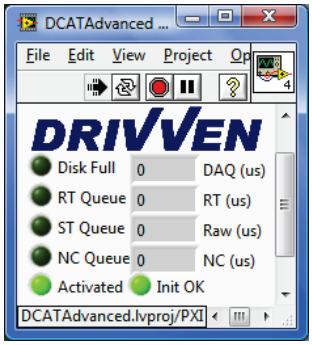
- The current tab in the main tab control controls the state of the real time application. The Measurement Setup and Engine Setup tabs (1st 2 tabs) place the real time application is a setup state and the rest of the tables place the real time application in the running state.
- When streaming to disk, the setup tabs are disables. This prevents the user from stopping the saving by accidently selecting the wrong tab.
- When saving a summary file, all the tabs are disabled except the summary and saving tabs. This keeps the user from changing the calculations in the middle of a file.
- Some calculations need to execute every loop (strip chart) and others only need to execute when they are visible (table of current values). The calculation that only need to execute when visible may be placed in the case structure based on the current tab. The calculations that need to execute every time need to be placed outside the case structure based on the current tab.
- Calculated results are used on multiple tabs and in strip charts that always need to update so they need to be placed off the tab control so that they will continually update.
- Before the user interface can run, the "Connected" CalPoint needs to be updated 2x to ensure that all the CalPoint values have been updated once. If not, the initialization may happen with bad values.

Running the DCATAdvanced Example

The DCAT Advanced example should be copied to a new location before changing any of the files. Failure to move the files may result in custom changes being overwritten when upgrading to newer version of DCAT. The DCAT advanced example is located at “C:\Program Files\Drivven\DCAT\DCATAdvanced”. The folder contains the example project “DCATAdvanced.lvproj” which is set up to run DCAT on a PXI chassis with a PXI-7831R in slot 2. The PXI-7831R is set up to have a NI-9151 on connector 1 with a 9411 in slot 1. This should be change to match the desired hardware setup.

Starting DCATAdvanced Example

1. Start the DCATAdvanced project.
2. Verify that the IP address of the PXI chassis is correct.
3. Open the “DCATAdvanced FPGA” VI and compile it if needed.
4. Open the “DCATAdvanced Target Program” VI and verify that there are no errors.
5. Run the CalVIEW Master VI on the block diagram of the “DCATAdvanced Target Program” VI. The CalVIEW Master VI should always run before deploying the target. Please refer to the CalVIEW user’s manual for more details on CalVIEW.
6. Run the “DCATAdvanced Target Program” VI. The **Activated** and **Init OK** indicators should turn green.



   a. If the **InitOK** light remains red, please run the CalVIEW Master VI and try again.
   b. If the **Activated** indicator remains red, please check your activation. If the **Activated** light continues to fail, please contact Drivven.
7. Start the CalVIEW consol. Start→Programs→Drivven→CalVIEW→CalVIEW.
8. Setup the CalVIEW consol.
   a. File→Select Target Machine to open the target selection dialog box. Select or enter the IP address of the target.
   b. File→Select Host VI and select the “DCATAdvanced User Interface” VI.
   c. File→Load Pairing File and select the “DCATAdvanced Paring.par” file.
9. Verify that the pairing has no errors and start the user interface.

Measurement and Engine Setup

Before collecting and analyzing data, the Measurement Setup, the Medium Speed DAQ Setup, and the Engine Setup tabs must be configured. While on these setup tabs, no data is collected or processed. Data collection and processing will automatically begin when the user is on any of the following tabs.

To see a description of a control or indicator, hover over the parameters with the mouse cursor to see the tip strip or context-help window message. The context help window may be accessed by pressing ctl-H.

The DCAT user interface also provides a sidebar of information to the right of the tabs that is always visible. The sidebar shows status information such as engine speed and data queue statistics that are useful at all times.

Measurement Setup

The Measurement Setup tab, shown in Figure 40, contains all of the configuration information for each measurement channel. The total number of measurement channels (**SyncChannelCount** and **AsyncChannelCount**) should be specified first. For each measurement, a physical channel, type, scaling, and associated cylinder should be set. The **PhysicalChannel_SyncX** is the device and channel name recognized by DAQmx. Ex. “Dev1\ai0”. The **MeasurementType_SyncX** allows the program to separate the specific measurements that are used for the calculations. A type of “undefined” should be used if the measurement type is not listed. Several options are available for the scaling of a measurement. Many of the defined measurement types require that the scaling factors be in terms of a specific engineering unit. The **AssociatedCylinder_SyncX** check boxes determine which cylinders the channel should be associated with. Some measurements, like cylinder pressure, should only be associated with a single cylinder. Others may be associated with multiple cylinders, such as manifold pressure. Please note that associating measurements with multiple cylinders may slow down the system. The **MeasurementName** is used to store the data in the file and for display. The units of undefined measurement types may be set to a text string that is used for display.

The **Filter_SyncX** refers to 1 of 16 filters that are defined on the Filtering tab. Each measurement may be set to the same filter or to its own filter.

Each measurement may be disabled with the **Disable_SyncX** button. When disabled, no data is recorded on that measurement and the configuration information for that measurement is not included in the saved files.

The combustion analysis calculation sets may be enabled and disabled depending on the specific requirements of the testing program. Please note that some of the calculations from one group may be required as inputs to another group. For example, heat release with heat transfer requires the cylinder temperature calculations.

PressureMetrics: Enables the calculations contained in the Pressure Metrics tab.
MEP: Enables calculation of Mean Effective Pressures (GMEP, PMEP, NMEP). The MEP calculations are displayed on the Summary tab.
CylTemp: Enables the calculation of estimated cylinder temperature for the heat-transfer calculations. This group must be enabled if heat release is calculated with heat transfer.
HeatRelease: Enables the calculations contained on the Heat Release tab.
Misfire: Enables the calculations related to misfire detection. Misfire detection is reported on the Summary tab.
Knock: Enables the calculations for the engine knock calculations.
Noise: Enables the calculation of combustion noise.

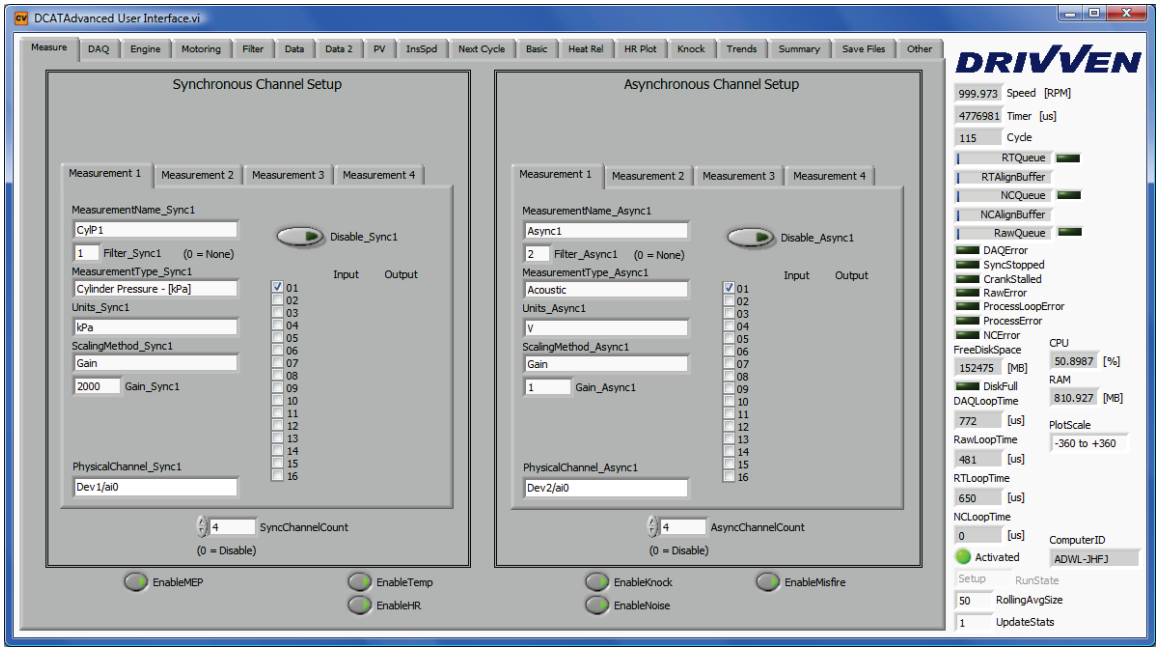


Figure 40: Measurement Setup Tab

Medium Speed / FPGA DAQ Setup

The Medium Speed DAQ Setup tab, shown in Figure 41, contains all the setup parameters needed to collect medium speed data. The tab control contains an easy to use XControl designed to allow the user to configure all properties. The XControl contains 3 parts: the sample rate, measurement editing, and the summary of measurements. The user may select a measurement from the summary table to edit. Once selected, the measurement configuration may be altered in the editing section. New measurements may be added using the **Insert** button. Selected measurements may be removed using the **Delete** button. The order of the measurements may be changed using the **Move Up** and **Move Down** buttons.

The Medium Speed DAQ is set up to scale measurements based on a gain and offset or a standard thermocouple calibration. Data is saved to the raw data file as a raw voltage for gain and offset scaling or as the temperature for thermocouples. The **Signal Max** and **Signal Min** values should represent the range of voltages for gain and offset scaling and the range of temperatures for thermocouples.

The **Sample Rate** of all measurements must be the same and must not exceed the sample rate of any device. Although the sample rate may be set as high as the modules will allow, large sample rates will require use more memory and CPU than modest sample rates. Each channel has the option of using a point-by-point second order Butterworth filter. The **Filter Cutoff** frequency must be less than half the sample rate. A **Filter Cutoff** frequency of 0 turns the filter off.

The FPGA DAQ Setup tab, shown in Figure 41, contains all the setup parameters needed to collect display FPGA data. The tab control contains an easy to use XControl designed to allow the user to configure all properties. All FPGA Data in the FIFO will be recorded in the raw data file regardless of the setup. However, only the data corresponding to a specified ID number in the setup XControl will be displayed, averaged, and saved in the summary data file. The XControl contains 2 parts: measurement editing, and the summary of measurements. The user may select a measurement from the summary table to edit. Once selected, the measurement configuration may be altered in the editing section. New measurements may be added using the **Insert** button. Selected measurements may be removed using the **Delete** button. The **Sort** buttons may be used to order the measurements by ID number.

The FPGA FIFO data is set up to scale measurements based on a gain and offset. Care should be taken when scaling the data. The FPGA Data only supports integer formats and therefore may require scaling a bit count to engineering units.

The sample rate of the FPGA data is determined in the FPGA. Each ID may have its own sample rate.

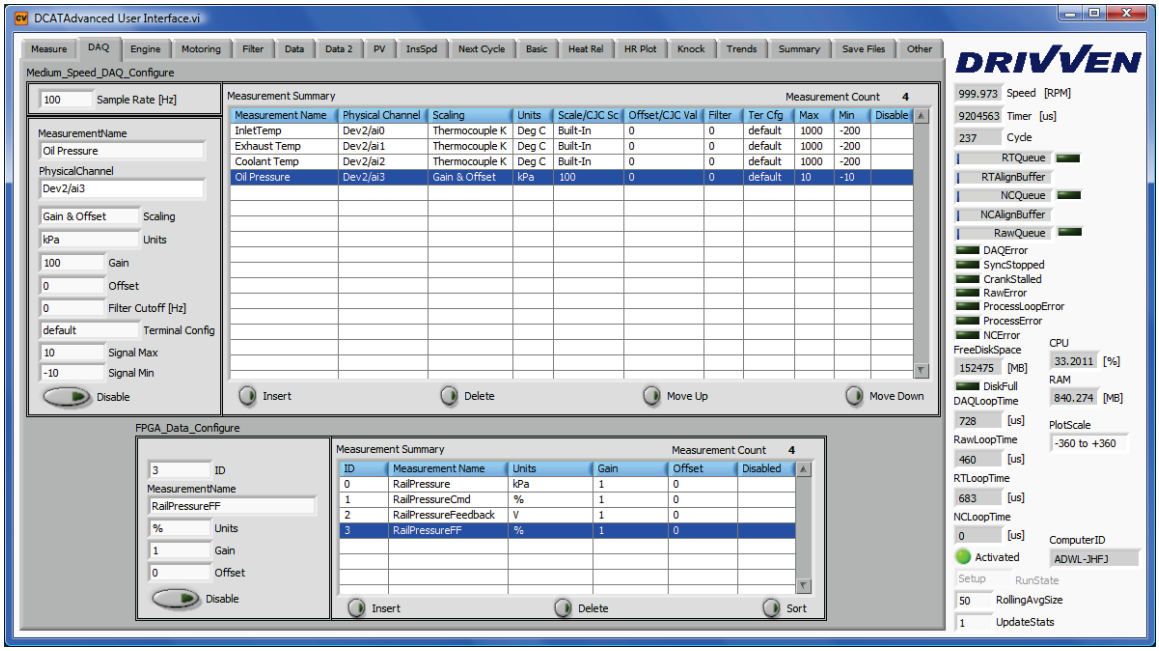


Figure 41: Medium Speed / FPGA DAQ Setup Tab

Engine Setup

The Engine Setup tab, shown in Figure 42, is a collection of engine geometry parameters. The **EngineName**, **ProjectName**, and **Operator** are strings that are saved in the output file for reference. They are not use in the program. The engine geometry and valve timings are used throughout the program for various calculation sets and should always be fully specified. **HeadArea** and **CrownArea** parameters are entered as percentages of the bore area. For example, **HeadArea** and **CrownArea** set to 100% assumes that the head portion of the combustion chamber and piston top are flat, and meet at TDC. **HeadArea** and **CrownArea** set to 120% assumes that those surfaces have 20% more surface area than the bore area. These parameters should only be adjusted above 100% if heat release is calculated using heat transfer.

The **EncoderResolution** and **TDC_Offset** must be defined for DCAT to function properly. **TDC_Offset** must be in terms of whole encoder pulses. Please note that **TDC_Offset** is a user specified value and the Actual_TDC_Offset is the closest possible value. To set a partial encoder pulse TDC offset, use the **TDC_Fine_Adj** on the motoring tab. The **EncoderExtrapLevel** needs to match the EPT VI used in the FPGA. The DCATAdvanced example is set up with the 2, 4, 6, and 8 bits of extrapolation VIs in the FPGA. The current EPT is set by the **EncoderExtrapLevel** and defaults to 2 if an invalid number is used. The final resolution sampled by the DAQ system is displayed as the **SampleResolution**. The **SampleResolution** may be modified using the **EncoderExtrapEnable**, **EncoderExtrapLevel**, and **EncoderDivide**. **EncoderDivide** always affects the **SampleResolution** and should not be less than 1. However, the **EncoderExtrapLevel** only affects it when **EncoderExtrapEnable** is true. If the encoder setup results in a fractional sample resolution, the **EncoderResolution** is used as the **SampleResolution**.

The asynchronous sampling resolutions are set by the **AsyncRate** control. The **MaxSpeed** is used to determine if the encoder signal recorded with the asynchronous measurements needs to be reduced to a lower resolution. The resolution may be divided to ensure that every encoder pulse is clearly distinguishable at the maximum engine speed.

Misfire detection monitors the absolute peak combustion pressure or the **NMEP**. The misfire calculations must be enabled by pressing the **MisFireCalcs** button on the Measurement Setup tab. Each option for **MisfireMethod** is described below:

Peak Pressure (Absolute): Misfire is detected when the measured peak pressure falls below the specified **PressureThreshold**.
NMEP (Absolute): Misfire is detected when the calculated **NMEP** falls below the specified **PressureThreshold.**
Peak Pressure (% of Historic): Misfire is detected when the measured peak pressure falls below a **PercentThreshold** of a rolling average of the measured peak pressure. **HistoryLength** specifies the number of cycles to use for the rolling average.
NMEP (% of Historic): Misfire is detected when the calculated **NMEP** falls below a **PercentThreshold** of a rolling average of **NMEP**. **HistoryLength** specifies the number of cycles to use for the rolling average.

Next cycle control provides a framework for implementing next cycle control using DCAT. This becomes particular useful when DCAT is implemented in the same system as an engine controller. Please contact Drivven for more details. Any of the calculated values may be used as feedback to the control. **NC_Enable** enables the buffering and calculations required for the next cycle control. The **NC_Cutoff** specifies the angle to stop collecting data for each cycle.

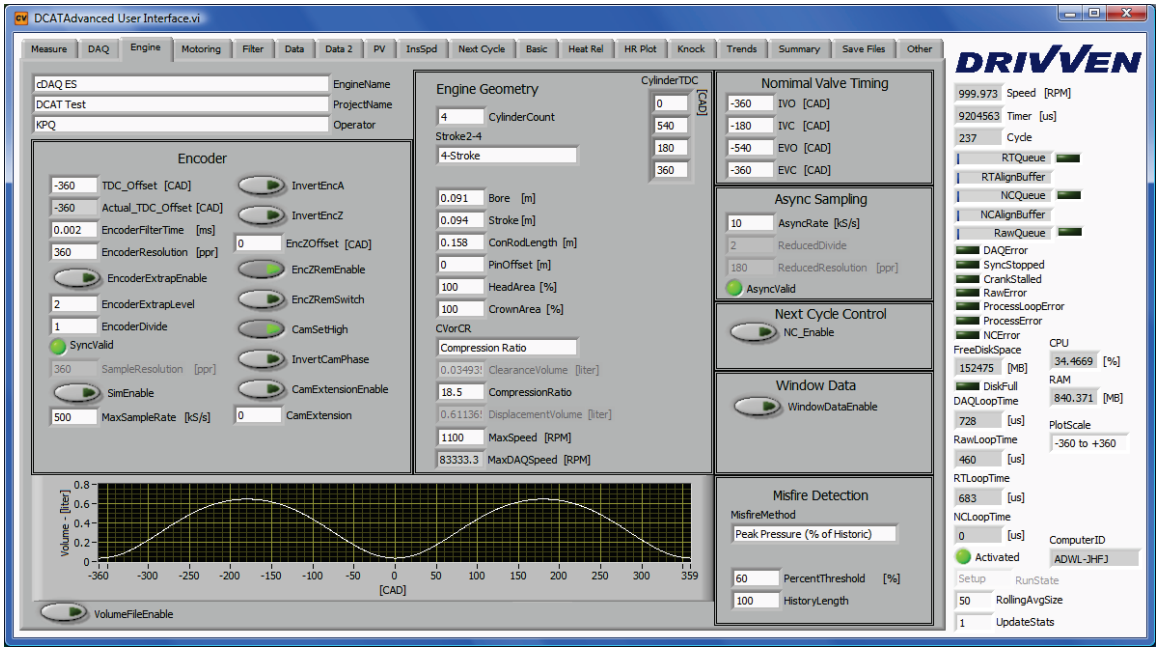


Figure 42: Engine Setup Tab

Motoring, TDC Alignment and Pegging

The motoring tab, shown in Figure 43, provides a place to check some of the key motoring parameters to ensure that the day to day variation of the engine is within acceptable ranges. Various motoring pressure metrics are displayed here. When the Motoring tab or any other tab to the right of the Engine Setup tab is selected, data acquisition begins according to the configuration of the first two tabs.

MotoringCylinder should be set to the cylinder number to be monitored. A **MotoringAverageSize** may be used to average a specified number of cycles for performing the calculations on this tab. Increasing this value will reduce the rate of updates.

The motoring polytropic coefficients of compression and expansion (**PolyC_motor** and **PolyE_motor**) are calculated here. The crank angle ranges to be used for calculating these parameters are entered on the Pressure Metrics tab. PolyC_motor and PolyE_motor are displayed graphically on the PV diagram. Tolerance ranges may be set for these parameters as a quick check of engine geometry and TDC setup.

When motoring, the **TDC_Check** method may be set to “Peak_Pressure”, “Spline Interpolation”, “Half Peak Pressure”, or Most Probable”. The **CalculatedTDCOffset** is displayed and the **TDC_Fine_Adj** parameter may be adjusted to achieve a **CalculatedTDCOffset** of zero. **TDC_Fine_Adj** may be use to correct partial encoder pulse offsets. However, setting **TDC_Fine_Adj** to a non-zero value may reduce the overall performance of the system because it causes a re-sampling of the measurement signals by linearly interpolating the arrays of data.

µDCAT provides several methods for pegging the cylinder pressure measurements. Pegging refers to dynamically adjusting the cylinder pressure offset. It is useful when using sensors that measure a dynamic pressure instead of an absolute pressure. Each selection of PeggingMethod is described below:

None: No pegging method is implemented and the measured pressure wave form, as calculated from the specified scaling parameters on the Measurement Setup tab, is used.
Constant: The average cylinder pressure (PressureAvg, not displayed) over a specified crank angle range is calculated. The center of the averaging range is specified by **PegLocation** in CAD. The PressureAvg is calculated over a range before and after **PegLocation** specified by **PegAverage** in CAD. The total angular range used to calculate PressureAvg is twice **PegAverage**. The difference between the specified **PegPressure** and PressureAvg (**PegPressure** – PressureAvg) is displayed in the **PeggingValue** table for each cylinder and is added to the entire pressure wave form for each cylinder.
Synchronous MAP: Synchronous MAP is identical to *Constant* except that a Manifold Absolute Pressure (MAP) sensor is sampled engine synchronously and averaged over the same angular range (MAPAvg, not displayed). MAPAvg is used instead of the *Constant*-mode user-specified **PegPressure**. **PegPressure** is not used. This method requires that a MAP sensor be connected to one of the measurement channels and configured on the Measurement Setup tab. It should be assigned to all relevant cylinders using the **AssociatedCylinders_SyncX** check boxes.
Polytropic: Uses the change in volume and pressure to determine the absolute pegging offset. This method calculates the pegging offset over the window specified for calculating the PolyC values on the Pressure Metrics page. The user must enter an accurate value for the polytropic coefficient of compression in **PegPolyC**. Please note that the TDC offset will affect the calculations because the cylinder volume is used.

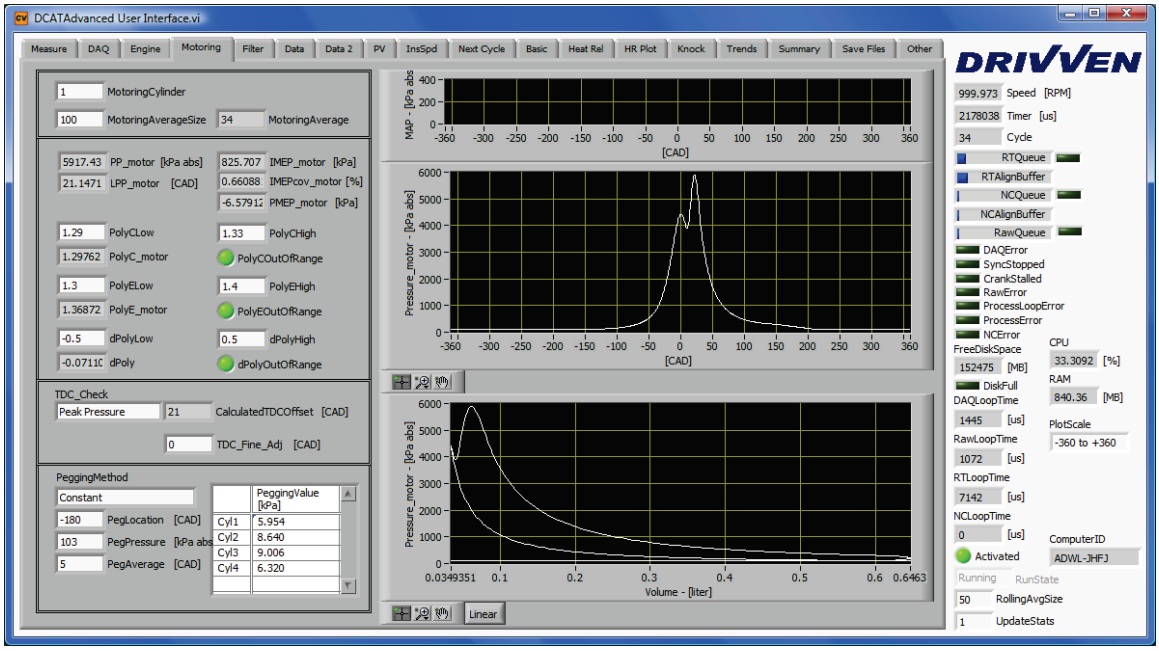


Figure 43: Motoring Tab

Filtering

The Filtering tab, shown in Figure 44, allows configuration of up to 16 filters. The filter number specified in the Measurement Setup tab for each measurement corresponds to the filter number configuration on this tab. There are several types of filters that may be selected with the **FilterMethodX** drop-down, including “Rolling N Cycles”, “Forward and Reverse IIR”, “FIR Filter with Rollback”, and “Boxcar”. A comparison of these filters is beyond the scope of this manual. When a filter type is selected, only the configuration parameters for that filter type will be shown. The filters are applied to the measurements before being displayed on all other plots.

An ensemble average of each measurement may also be used by pressing the **EnsembleEnable** button. When using the ensemble average, a specified number of cycles (**EnsembleSize**) are averaged at each crank angle and the averaged results are used in the calculations. Increasing **EnsembleSize** will reduce the rate at which the calculations are performed.

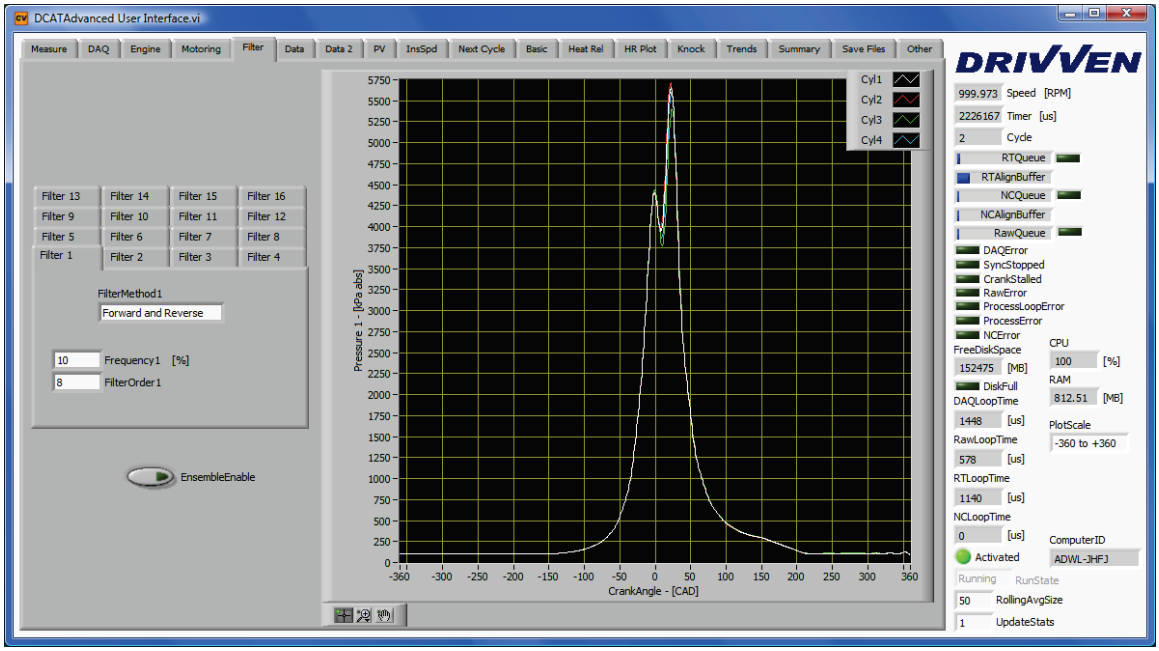


Figure 44: Filtering Tab

Viewing All Waveforms

The Data tabs, shown in Figure 45 through Figure 47, displays all the measurements associated with a single cylinder specified in **DisplayCylinder**. The data is aligned with TDC, scaled, and filtered. The undefined measurement types (selected by **MeasurementType_SyncX** on the Measurement Setup tab) are displayed on this tab according to the selected associated cylinder. This is the only tab where undefined measurement types are displayed.

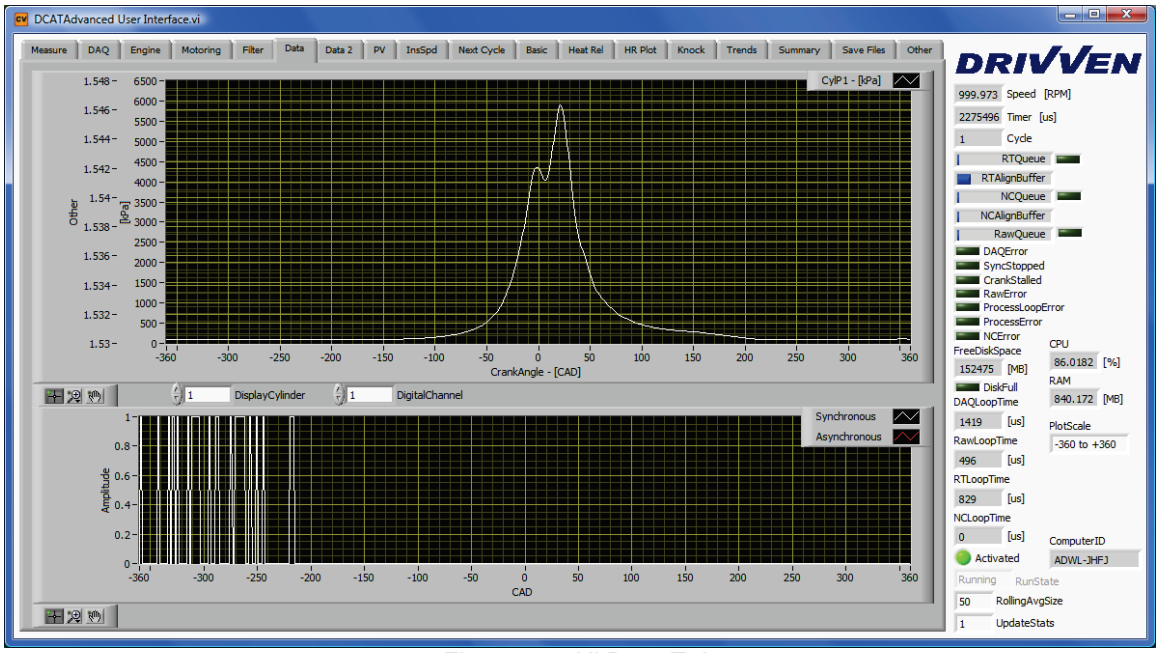


Figure 45: All Data Tab

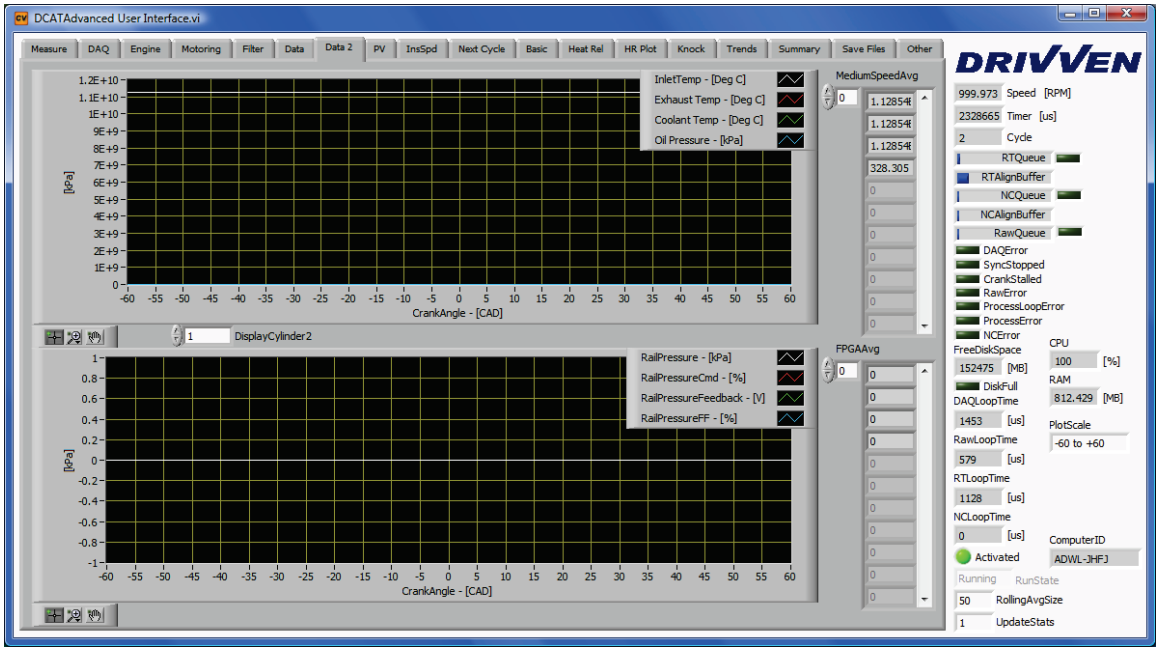


Figure 46: All Data Tab 2

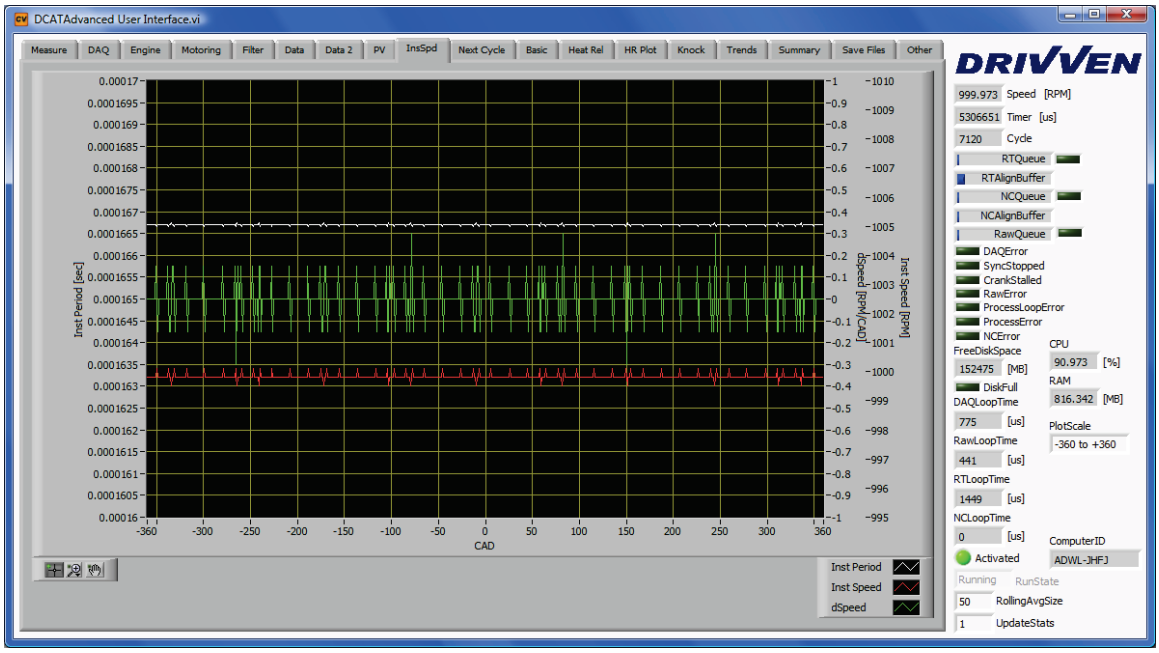


Figure 47: Instantaneous Speed Tab

PV Diagram

A full screen diagram, shown in Figure 48, may be helpful in general setup and alignment. The button at the lower left of the plot may be used to switch between logP-logV view and normal axis views. In the log view, the red lines represent the straight line generated from the polyC/PolyE values. The points on the red line show the PolyC/PolyE calculation windows.

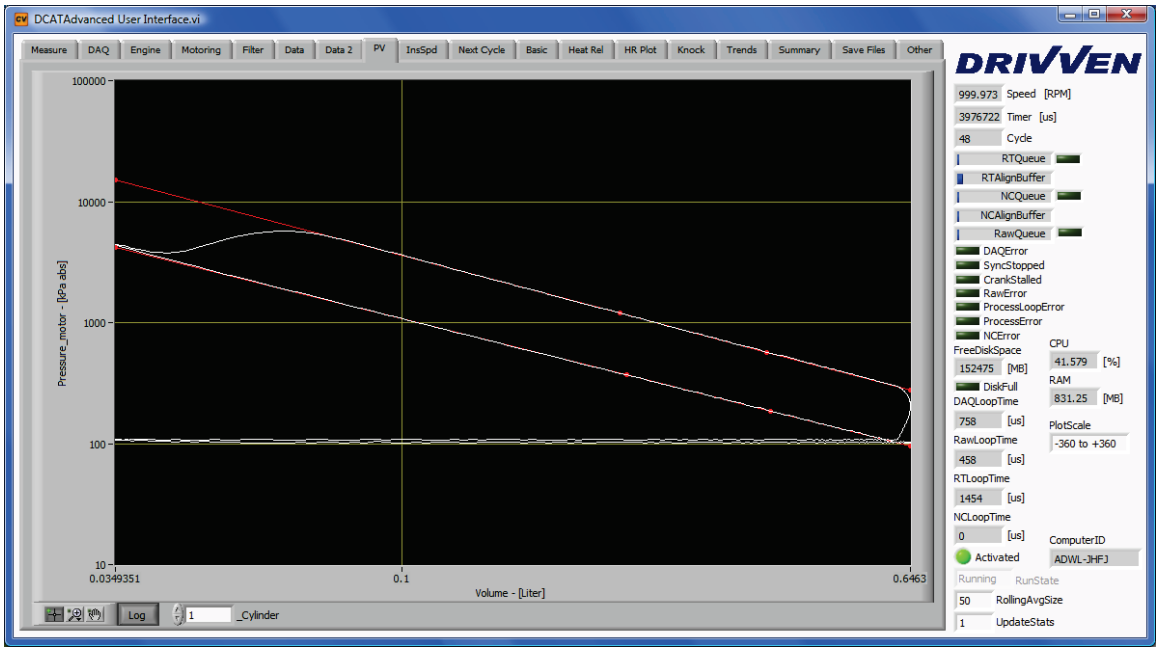


Figure 48: PV Diagram Tab

Next Cycle Control

The next cycle control tab, shown in Figure 49, was designed to provide a framework and example for next cycle control. The example is set up to calculate the pressure metrics and heat release using the same settings that are used for the normal DCAT calculations. The control strategy found in the example was designed to be a framework and is therefore not complete. The next cycle tab is divided into two sections. The left side of the tab is dedicated to calculating the parameters and verifying that the calculations complete in an appropriate time. The right side is dedicated to the implementation of the control strategy.

The next cycle control must be enabled on the engine setup tab. The **NC_Cutoff** (found on the engine setup page) and **NC_AlarmAngle** are used to specify the acceptable start and end angles of the calculation. The calculations should begin after the **NC_Cutoff** angle and will produce a true value in the **NC_Alarm** if not finished by the **NC_AlarmAngle**. The alarm angle is used to verify that the engine controller has enough time to apply the newly calculated value to the control strategy. The **NC_Watchdog** increments with the cycle number. If the engine controller should verify that the **NC_Watchdog** is incrementing occasionally or default to a safe state. A dial and plot show the estimated angles that the different steps in the process complete.

The framework for the control provides a table used to lookup the setpoint of the parameter based on speed and load. The calculated parameter has the option to filter the value using a simple weighted history and a PID controller for the engine as a whole or each cylinder individually.

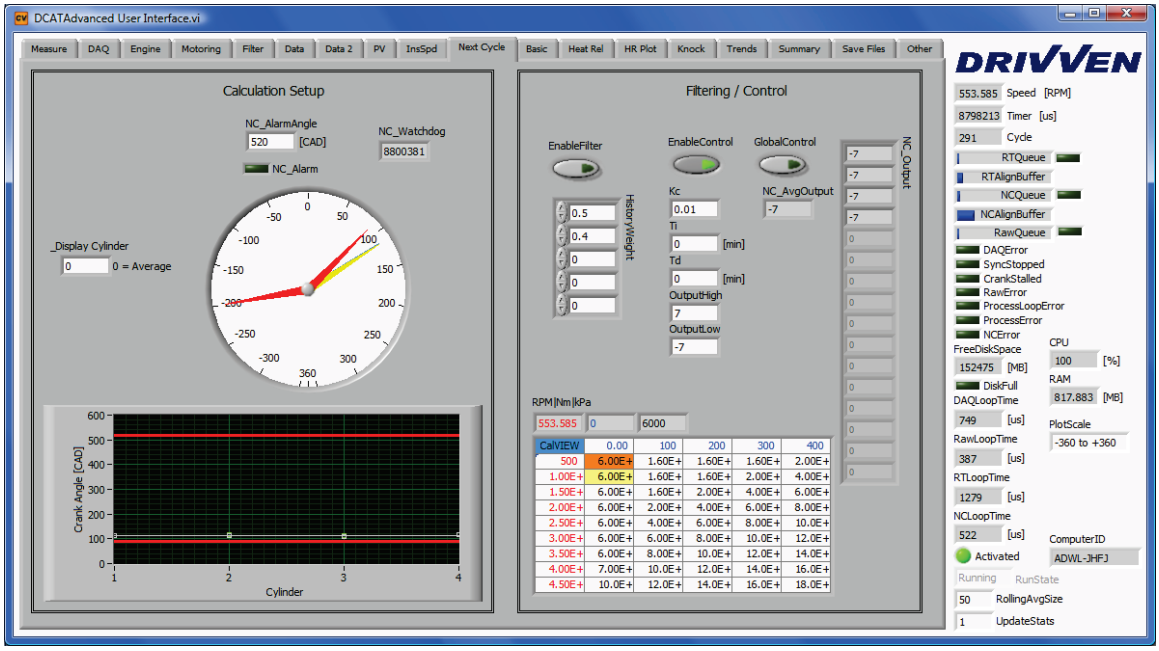


Figure 49: Next Cycle Control Tab

Calculation Setup and Results

Many of the combustion analysis calculations require additional setup. The additional setup parameters may be found on the following tabs. The individual numerical results are displayed in the tables, having columns for each cylinder. The arrays of angle-based data are displayed in plots. Each calculated table value has rolling statistics associated with it. The displayed statistics may be turned off by un-checking the control next to the table. Each individual plot can be configured for auto-scaling or manual scaling of the Y axis, by right-clicking on the scale and checking or un-checking the "AutoScale" menu item. There are three parameters on the bottom right side bar for configuring the Crank Angle Degrees (CAD) scale.

Pressure Metrics

The Pressure Metrics tab, shown in Figure 50, displays the results of basic analysis of each cylinder pressure wave form. The angle parameters at the top of the tab are required for calculating each pressure metric. The table-based calculated parameters are as follows:

PP [kPa]: Peak Pressure
LocPP [CAD]: Location of Peak Pressure
PolyC: Polytropic Coefficient of Compression
PolyE: Polytropic Coefficient of Expansion
MPR [kPa/CAD]: Maximum Pressure Rise
LocMPR [CAD]: Location of Maximum Pressure Rise
Intake [kPa]: Average pressure during intake valve opening
Exhaust [kPa]: Average pressure during exhaust valve opening

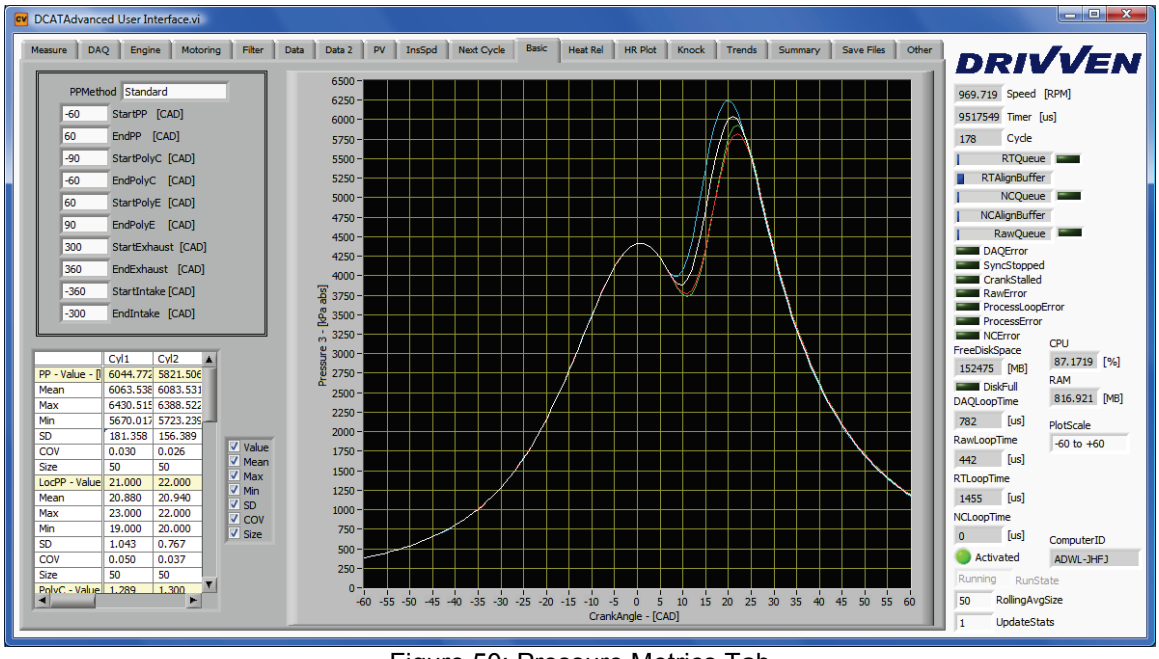


Figure 50: Pressure Metrics Tab

Heat Release

The Heat Release tabs, shown in Figure 51 and Figure 52, displays in table and plot format the Heat Release Rate (HRR) and Mass Fraction Burn (MFB) percentages. The dropdown menu **HeatReleaseMethod** selects one of several heat release methods. Different configuration parameters will be shown for each heat release method. The **StartMethod** and **EndMethod** define how the start and end of combustion are determined. The in-cylinder temperature calculation setup is also on this page. If **ResidualFraction** is zero, **IntakeTemp** and **IVC_Window** are used. The starting pressure is the average pressure ending at IVC with **IVC_Window** length. When using **ResidualFraction**, the **IntakeTemp** is adjusted depending on the **IntakePressure**, **ExhaustPressure** and **ExhaustTemperature**. For more details on the heat release calculations, please refer to the Calculations section of this manual.

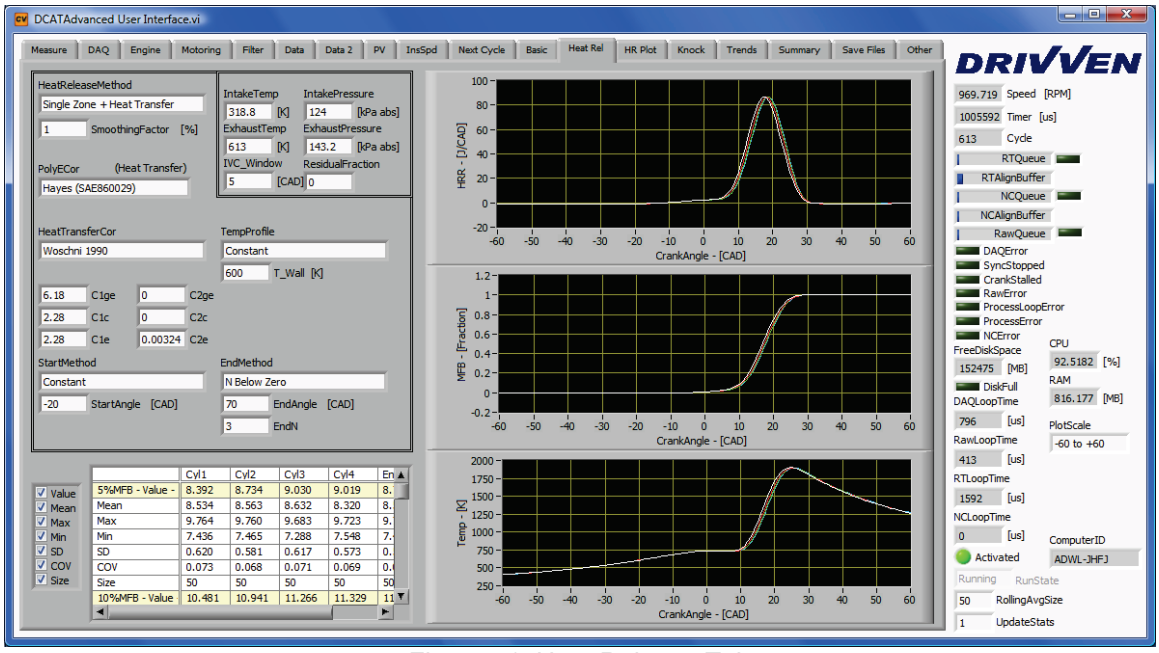


Figure 51: Heat Release Tab

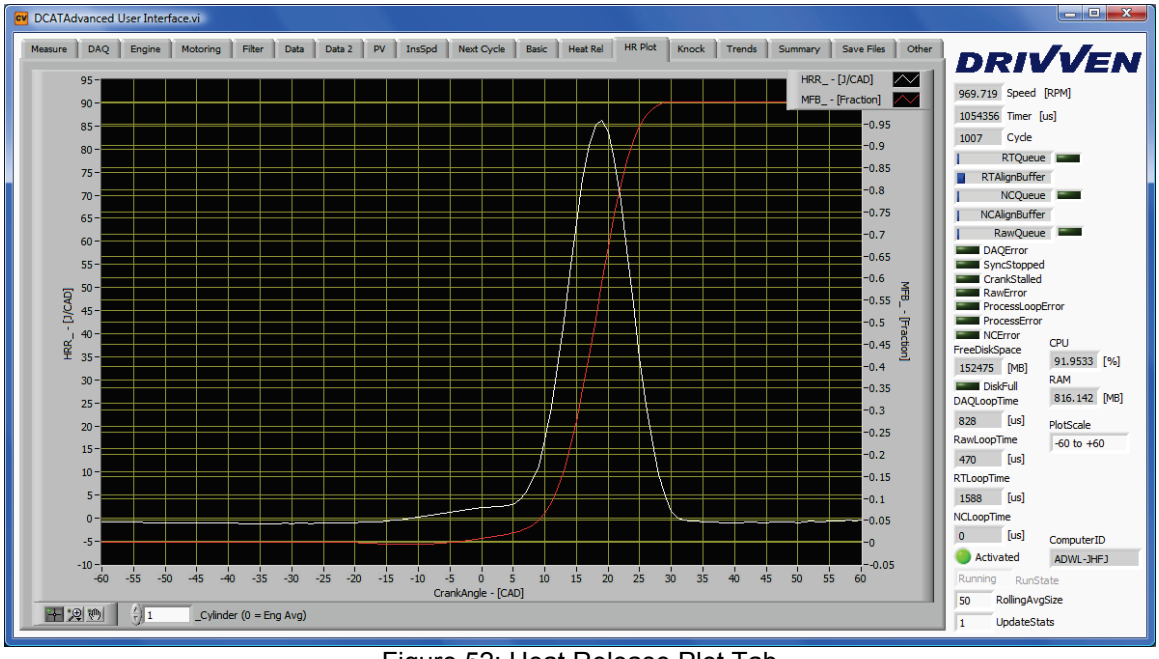


Figure 52: Heat Release Plot Tab

Knock

The knock tab, shown in Figure 53, may be used to calibrate and display knock. Knock is calculated using the Bandpass, Rectify, Integrate, Comair (BRIC) method.

The knock XControl is used to provide a visual display of the cylinder pressure and knock sensor signals so that the user can select a window for knock detection and a window for a reference (top left plot). An FFT of the windows is displayed so that the user may select a frequency window (top right plot). The filters are applied to the windows and another FFT is applied and the results are displayed (lower right plot). Finally, the windows are integrated and compared. The results are displayed for the user (lower left plot). The user can save and load data sets. Up to 8 data sets may be loaded and viewed. The calculated knock values are updated on each dataset when any of the windows change.

Knock [dB]: Calculated Knock
PtoPKnock []: Filtered Peak to Peak Value

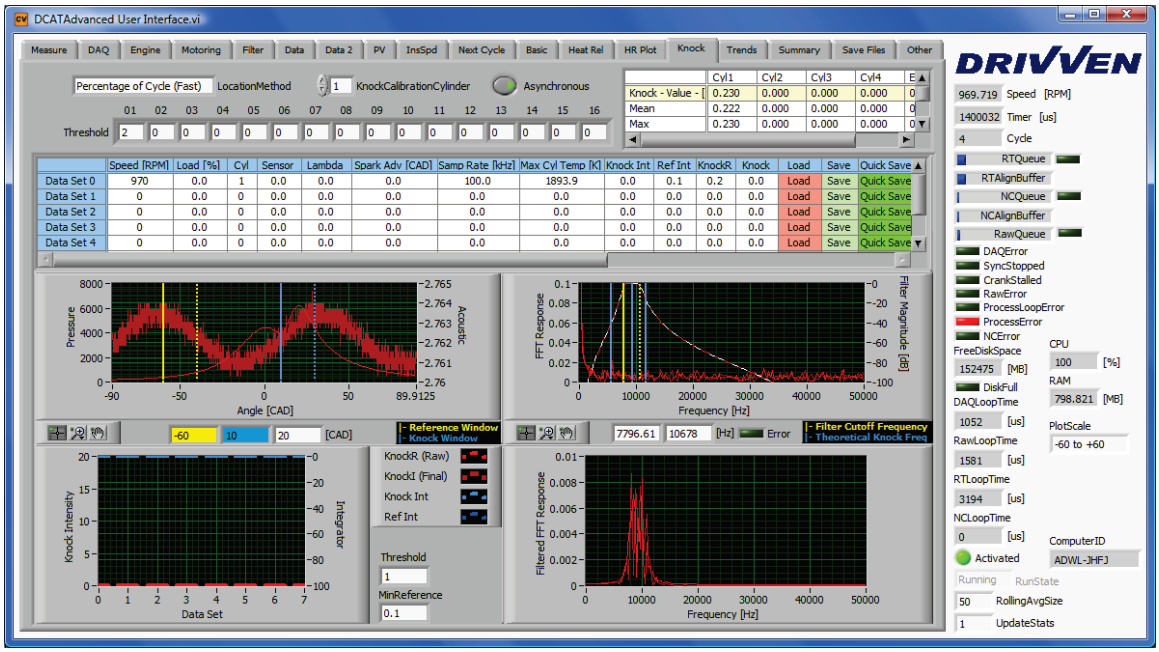


Figure 53: Knock Tab

The Knock XControl (Figure 54) is a graphical knock calibration tool. It can be used to the BRIC windows and filters coefficients. There are five parts of the tool.

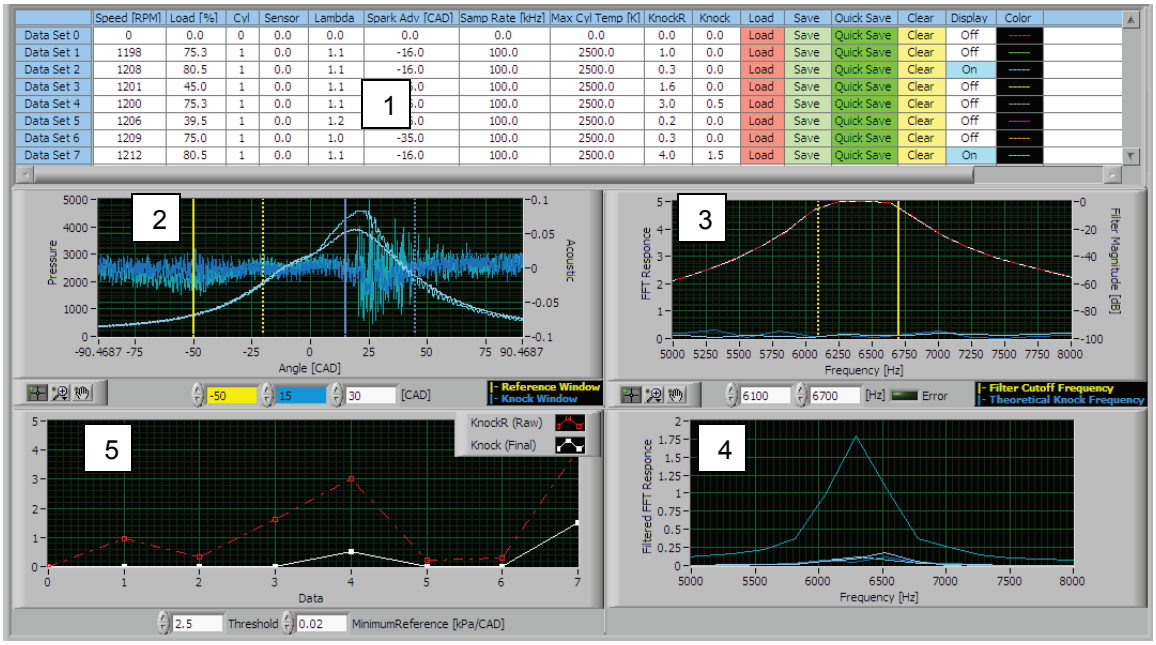


Figure 54: Knock XControl

1. The summary displays up to 8 data sets. Each data set may be saved or loaded from file. Pressing the save or load button opens a file dialog box. The clear button on the summary display clears the data set. The display button turns on an off the corresponding waveforms on the plots below. The color indicates the corresponding waveform color.
2. The window plot shows the pressure waveform overlaid with the acoustic waveform. This allows the user to view the ripple on the pressure waveform along with the corresponding acoustic signal. This also allows the user to check for other issues like valve noise and other cylinder knock events. The yellow window defines the reference window. Likewise, the blue window defines the knock window. The user may drag and drop the start cursor on both windows or enter the value in the appropriate colored box below.
3. The filter magnitude response plot displays the filter cutoff with an FFT of the knock and reference windows. The filter cutoffs are represented with cursors that the user may drag as needed.
4. The FFT of the filtered windows is displayed based on the current settings.
5. The integrated values of knock are displayed for each data set. The plots show the ratio of the integrated and filtered windows before and after the threshold is applied.

Viewing Trends and Summary Data

Trends

The Trends tab, shown in Figure 55, displays charts for a quick view of the historic data. Any calculated parameter can be selected to see the historical trends. The available values are displayed in the list box to the left of the trend. Holding Shift or Ctrl allows for selecting multiple values to trend. Changing the selection will reset the trends. The listed available trends may be filtered using the controls below the list box. Please note that trends are updated periodically. Therefore, not every cycle of data will be displayed in the trend.

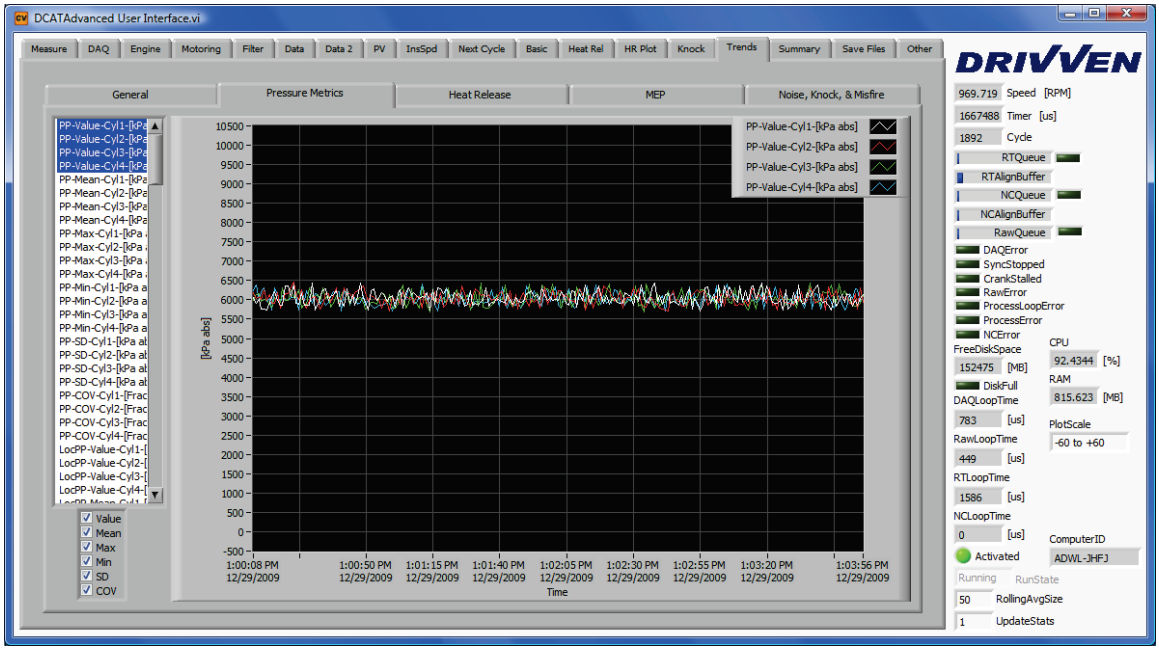
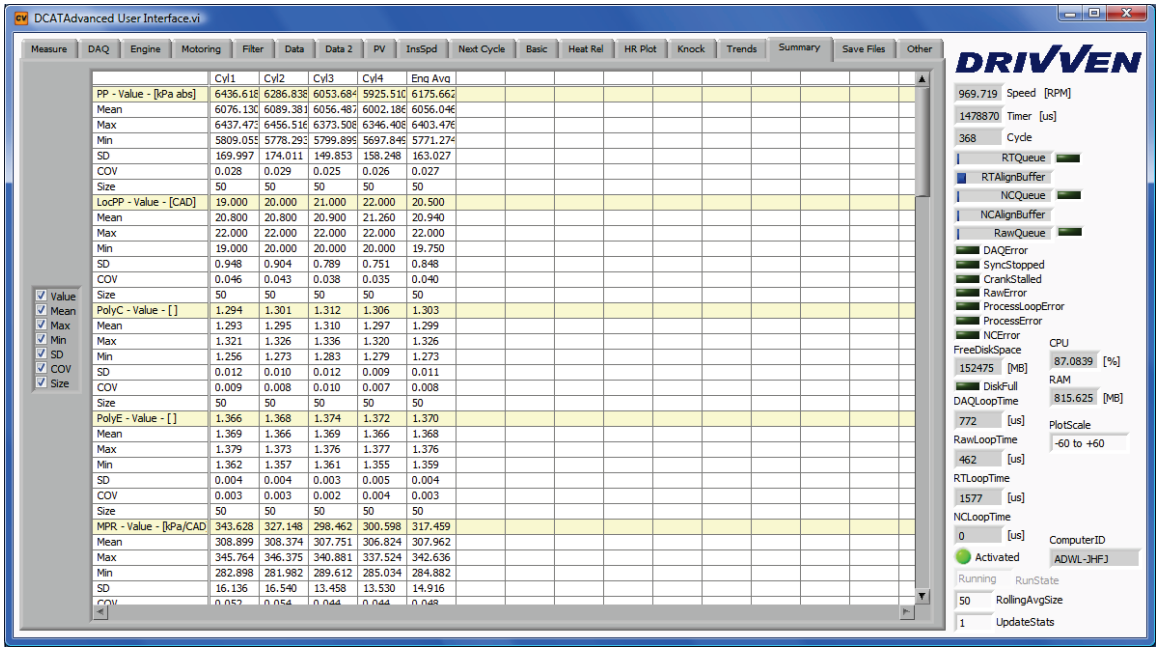


Figure 55: Trends Tab

Summary

The Summary tab, shown in Figure 56, displays a single table of all calculated values and associated rolling statistics. The values in the summary table are a collection of the same table values from the previous tabs. Please note that the Mean Effective Pressure (MEP) results and the misfire results are only displayed in the Summary tab table. Below is a definition of the MEP calculations:

IMEP: Indicated MEP over the firing portion of the cycle defined as -180 to 180 CAD.
PMEP: Pumping MEP over the portions of the cycle defined as -360 to -180 and 180 to 360 CAD. PMEP is a negative value.
NMEP: Net MEP resulting from IMEP + PMEP. NMEP is effectively over the complete cycle defined as -360 to 360 CAD.



	Cyl1	Cyl2	Cyl3	Cyl4	Eng Avg
PP - Value - [kPa abs]	6436.618	6286.838	6053.684	5925.510	6175.662
Mean	6076.130	6089.381	6056.487	6002.186	6056.046
Max	6437.473	6456.516	6373.508	6346.408	6403.476
Min	5809.055	5778.293	5799.895	5697.845	5771.274
SD	169.997	174.011	149.853	158.248	163.027
COV	0.028	0.029	0.025	0.026	0.027
Size	50	50	50	50	50
LocPP - Value - [CAD]	19.000	20.000	21.000	22.000	20.500
Mean	20.800	20.800	20.900	21.260	20.940
Max	22.000	22.000	22.000	22.000	22.000
Min	19.000	20.000	20.000	20.000	19.750
SD	0.948	0.904	0.789	0.751	0.848
COV	0.046	0.043	0.038	0.035	0.040
Size	50	50	50	50	50
PolyC - Value - [ ]	1.294	1.301	1.312	1.306	1.303
Mean	1.293	1.295	1.310	1.297	1.299
Max	1.321	1.326	1.336	1.320	1.326
Min	1.256	1.273	1.283	1.279	1.273
SD	0.012	0.010	0.012	0.009	0.011
COV	0.009	0.008	0.010	0.007	0.008
Size	50	50	50	50	50
PolyE - Value - [ ]	1.366	1.368	1.374	1.372	1.370
Mean	1.369	1.366	1.369	1.366	1.368
Max	1.379	1.373	1.376	1.377	1.376
Min	1.362	1.357	1.361	1.355	1.359
SD	0.004	0.004	0.003	0.005	0.004
COV	0.003	0.003	0.002	0.004	0.003
Size	50	50	50	50	50
MPR - Value - [kPa/CAD	343.628	327.148	298.462	300.598	317.459
Mean	308.899	308.374	307.751	306.824	307.962
Max	345.764	346.375	340.881	337.524	342.636
Min	282.898	281.982	289.612	285.034	284.882
SD	16.136	16.540	13.458	13.530	14.916
COV	0.052	0.054	0.044	0.044	0.048

Figure 56: Summary Tab

Saving Data

The Save Files tab, shown in Figure 57, provides configuration parameters for saving two types of data sets. The left side of the tab is used for saving raw data. “Raw Data” consists of the absolute voltages measured directly from each channel. The raw data is streamed to disk without any processing. Therefore, the data is not scaled or aligned to the cylinder. However, every configuration parameter from the setup and calculation tabs is also saved to the raw data file so the test environment can be recreated in post processing.

The right side of the tab is used for saving summary data. “Summary Data” is the set of values calculated every cycle, such as pressure metrics and heat release. The statistics are also calculated and saved in the file. The ensemble average of the scaled and aligned waveforms is also saved with the summary data.

Raw data and summary data may be recorded simultaneously. The cycle number and time stamp, which is recorded in both file types, can be used to align the summary data with the raw data.

Both raw data and summary data types have identical file saving configurations. The raw data file saving parameters will be described below.

Raw_Path: Defines the full path to the directory for storing the file
Raw_Filename: Defines the name of the raw data file. A suffix of XXX tells µDCAT to append a file number to the end of the name. The file number will be assigned and incremented according to **Raw_StartCounter**, **Raw_Counter** and **Raw_ResetCounter**.
Raw_SaveMethod: Selects the method for determining the end of saving raw data to file. The options are as follows:

- **Continuous:** Continuously save data while the **Start/StopRawFile** button is pressed.
- **N-Cycles:** Saves data for number of cycles specified by **Raw_MaxCycles.**
- **Timed:** Saves data for a time specified by **Raw_MaxTime**.
- **Size:** Saves data until a file size specified by **Raw_MaxSize** is reached.

Raw_Comments: String inserted into the raw data file.
Raw_StartCounter: Specifies the starting file number to be filled in for “XXX” in the filename.
Raw_ResetCounter: Resets the file number to **Raw_StartCounter**
Start/StopRawFile: Starts and stops the file saving process.

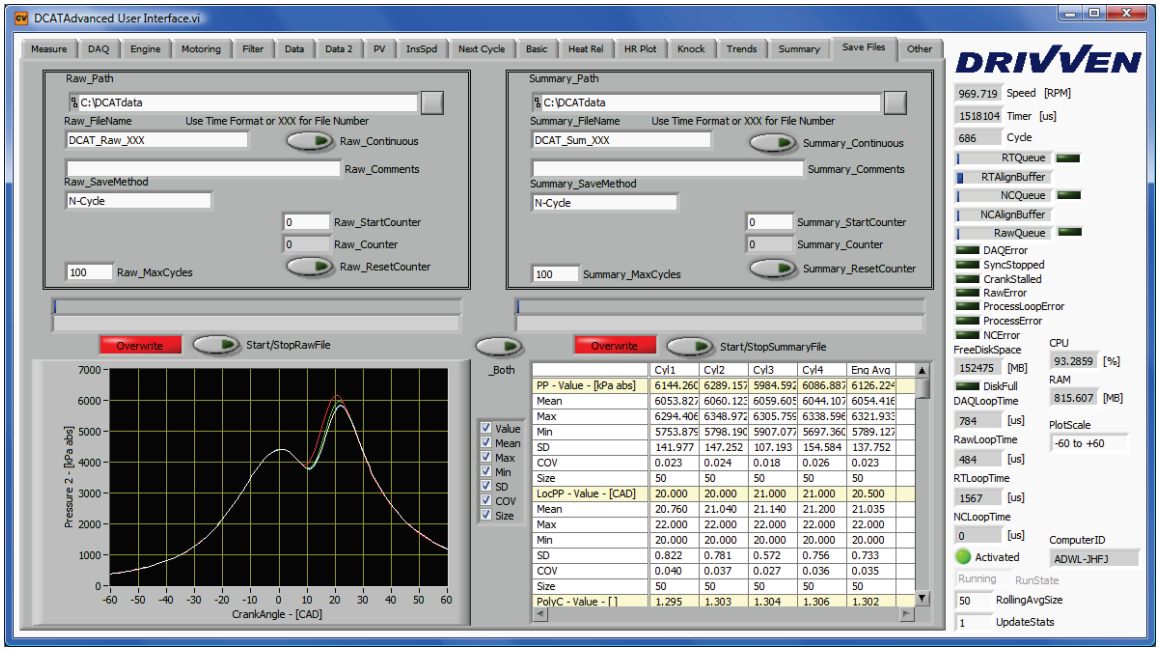


Figure 57: Saving Data Tab

Other

The other tab contains the text descriptions of errors. This tab also contains the setup controls for the CalVIEW Trends and Scopes. Please see the CalVIEW manual for more details on these features.

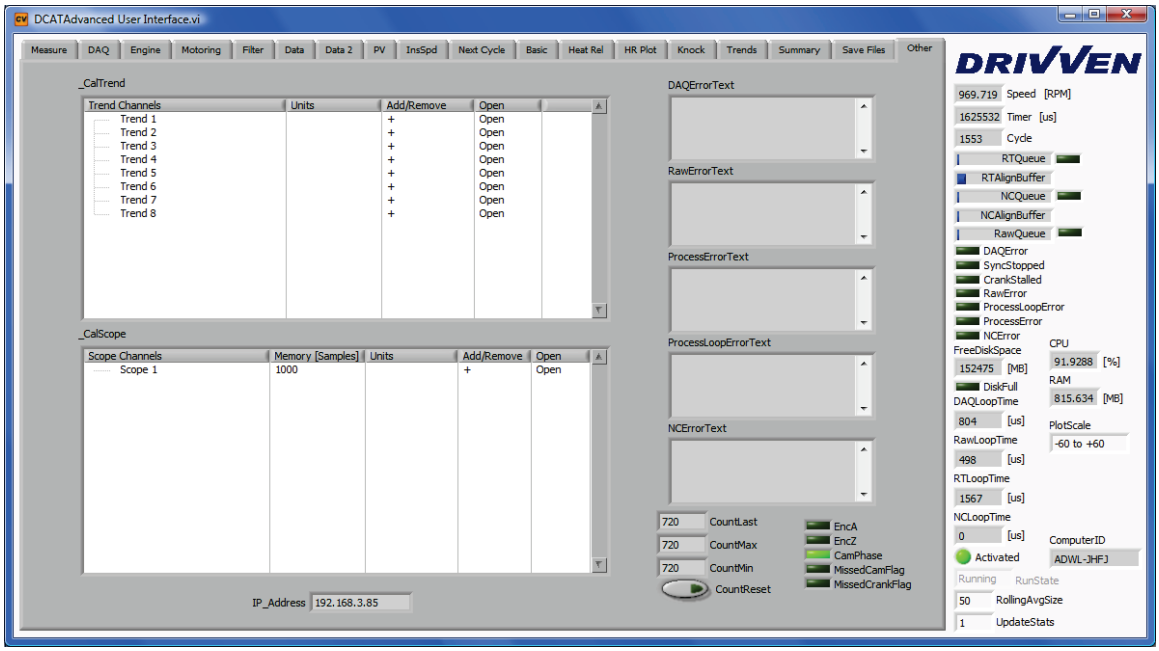
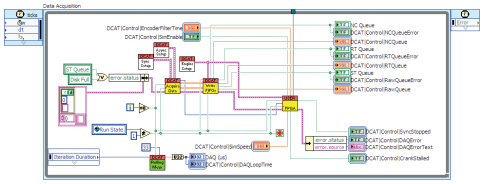
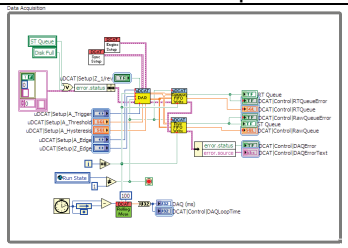
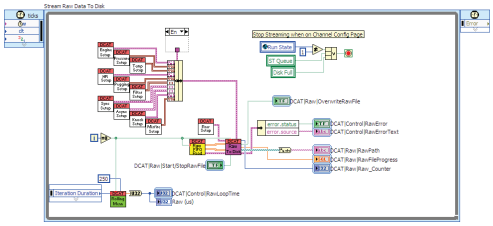
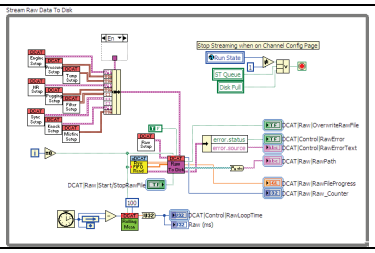
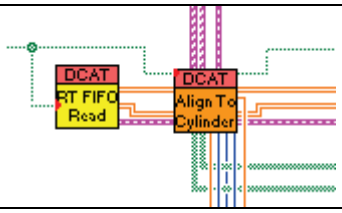
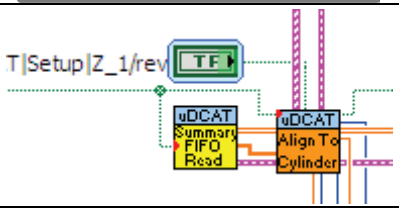


Figure 58: Other Tab

Overview of the uDCAT Example

The uDCAT example shows the full application converted to a uDCAT implementations. The differences between the Advanced and uDCAT application are listed below.

- Doesn't use an FPGA
  - Can only use an standard encoder
  - No advanced triggering options
- No asynchronous data
- No next cycle control support
- Target will only run on Windows
  - No real time support
- Doesn't collect digital data
  - Speed, time, and cycle are included together in an array
- Requires sever DCAT SubVIs to be replaced
  - Acquire Data
  - Raw and Summary FIFOs
  - Align to Cylinder
- No timed loops
  - Use while loops

	DCATAdvanced Example	uDCAT Example
DAQ Loop		
Stream Raw Data Loop		
Real Time Processing Alignment		

Creating a Post Processor Application

DCAT saves data in National Instruments' standard TDMS file format. TDMS files provide a powerful and flexible way to store and retrieve data. National Instruments provides many resources on their website to edit and view TDMS files outside LabVIEW in programs such as MatLab and Excel. The DCAT toolkit contains several VIs used to stream data from the TDMS files.

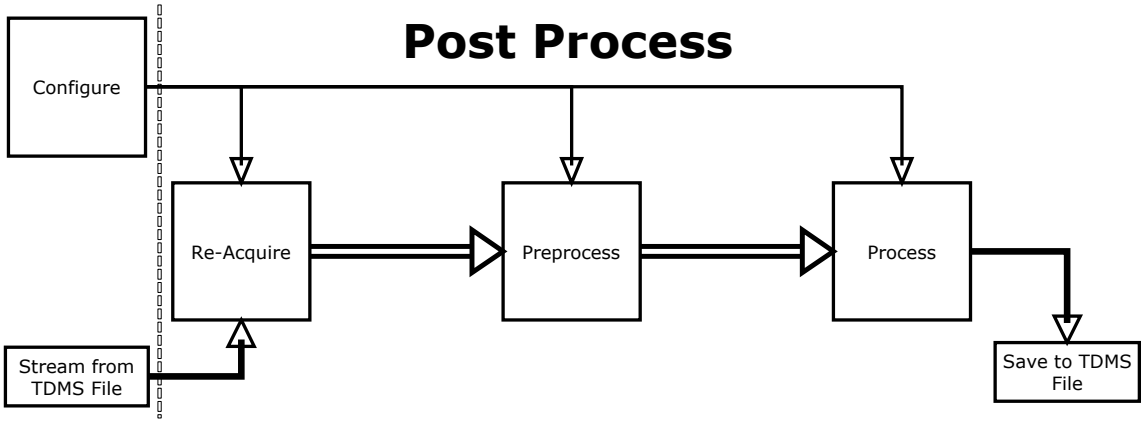
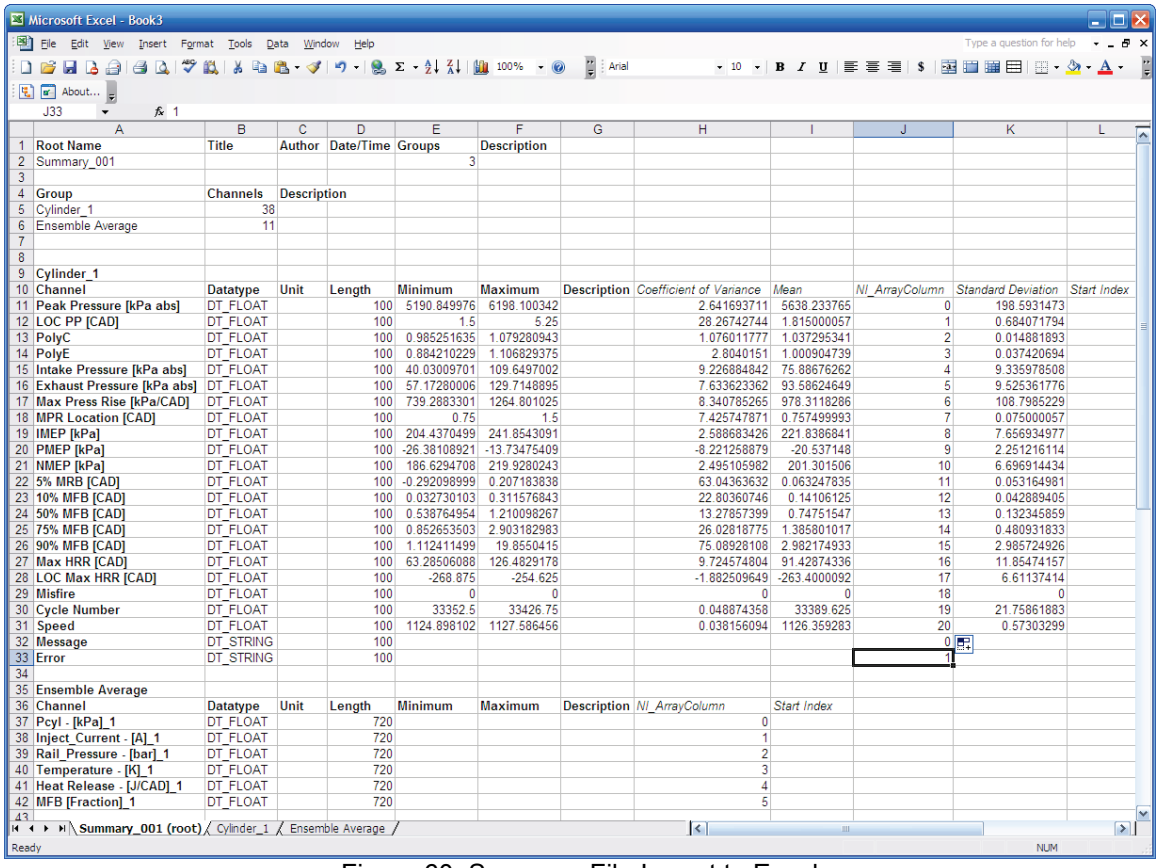


Figure 59 - Post Processing Overview

Opening TDMS Files in Excel

National Instruments provides a free excel plug-in that imports TDMS files. Importing a file creates a multi-page spreadsheet with all the data stored in the file. The first page contains a summary of all the groups and channels. Each group has a page where each column is a channel of data. Please note that Excel 2003 is limited to 256 columns and 65536 rows. Therefore, the custom properties (displayed in the first 2 rows on the main page) which contain all the calculation configuration information may need to be turned off. Raw data channels may also be too long and will be truncated.



Microsoft Excel - Book3

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Root Name**	**Title**	**Author**	**Date/Time**	**Groups**	**Description**						
2	Summary_001				3							
3												
4	**Group**	**Channels**	**Description**									
5	Cylinder_1	38										
6	Ensemble Average	11										
7												
8												
9	**Cylinder_1**											
10	**Channel**	**Datatype**	**Unit**	**Length**	**Minimum**	**Maximum**	**Description**	*Coefficient of Variance*	*Mean*	*NI_ArrayColumn*	*Standard Deviation*	*Start Index*
11	**Peak Pressure [kPa abs]**	DT_FLOAT		100	5190.849976	6198.100342		2.641693711	5638.233765	0	198.5931473	
12	**LOC PP [CAD]**	DT_FLOAT		100	1.5	5.25		28.26742744	1.815000057	1	0.684071794	
13	**PolyC**	DT_FLOAT		100	0.985251635	1.079280943		1.076011777	1.037295341	2	0.014881893	
14	**PolyE**	DT_FLOAT		100	0.884210229	1.106829375		2.8040151	1.000904739	3	0.037420694	
15	**Intake Pressure [kPa abs]**	DT_FLOAT		100	40.03009701	109.6497002		9.226884842	75.88676262	4	9.335978508	
16	**Exhaust Pressure [kPa abs]**	DT_FLOAT		100	57.17280006	129.7148895		7.633623362	93.58624649	5	9.525361776	
17	**Max Press Rise [kPa/CAD]**	DT_FLOAT		100	739.2883301	1264.801025		8.340785265	978.3118286	6	108.7985229	
18	**MPR Location [CAD]**	DT_FLOAT		100	0.75	1.5		7.425747871	0.757499993	7	0.075000057	
19	**IMEP [kPa]**	DT_FLOAT		100	204.4370499	241.8543091		2.588683426	221.8386841	8	7.656934977	
20	**PMEP [kPa]**	DT_FLOAT		100	-26.38108921	-13.73475409		-8.221258879	-20.537148	9	2.251216114	
21	**NMEP [kPa]**	DT_FLOAT		100	186.6294708	219.9280243		2.495105982	201.301506	10	6.696914434	
22	**5% MRB [CAD]**	DT_FLOAT		100	-0.292098999	0.207183838		63.04363632	0.063247835	11	0.053164981	
23	**10% MFB [CAD]**	DT_FLOAT		100	0.032730103	0.311576843		22.80360746	0.14106125	12	0.042889405	
24	**50% MFB [CAD]**	DT_FLOAT		100	0.538764954	1.210098267		13.27857399	0.74751547	13	0.132345859	
25	**75% MFB [CAD]**	DT_FLOAT		100	0.852653503	2.903182983		26.02818775	1.385801017	14	0.480931833	
26	**90% MFB [CAD]**	DT_FLOAT		100	1.112411499	19.8550415		75.08928108	2.982174933	15	2.985724926	
27	**Max HRR [CAD]**	DT_FLOAT		100	63.28506088	126.4829178		9.724574804	91.42874336	16	11.85474157	
28	**LOC Max HRR [CAD]**	DT_FLOAT		100	-268.875	-254.625		-1.882509649	-263.4000092	17	6.61137414	
29	**Misfire**	DT_FLOAT		100	0	0		0	0	18	0	
30	**Cycle Number**	DT_FLOAT		100	33352.5	33426.75		0.048874358	33389.625	19	21.75861883	
31	**Speed**	DT_FLOAT		100	1124.898102	1127.586456		0.038156094	1126.359283	20	0.57303299	
32	**Message**	DT_STRING		100						0		
33	**Error**	DT_STRING		100						1		
34												
35	**Ensemble Average**											
36	**Channel**	**Datatype**	**Unit**	**Length**	**Minimum**	**Maximum**	**Description**	*NI_ArrayColumn*	*Start Index*			
37	**Pcyl - [kPa]_1**	DT_FLOAT		720				0				
38	**Inject_Current - [A]_1**	DT_FLOAT		720				1				
39	**Rail_Pressure - [bar]_1**	DT_FLOAT		720				2				
40	**Temperature - [K]_1**	DT_FLOAT		720				3				
41	**Heat Release - [J/CAD]_1**	DT_FLOAT		720				4				
42	**MFB [Fraction]_1**	DT_FLOAT		720				5				
43												

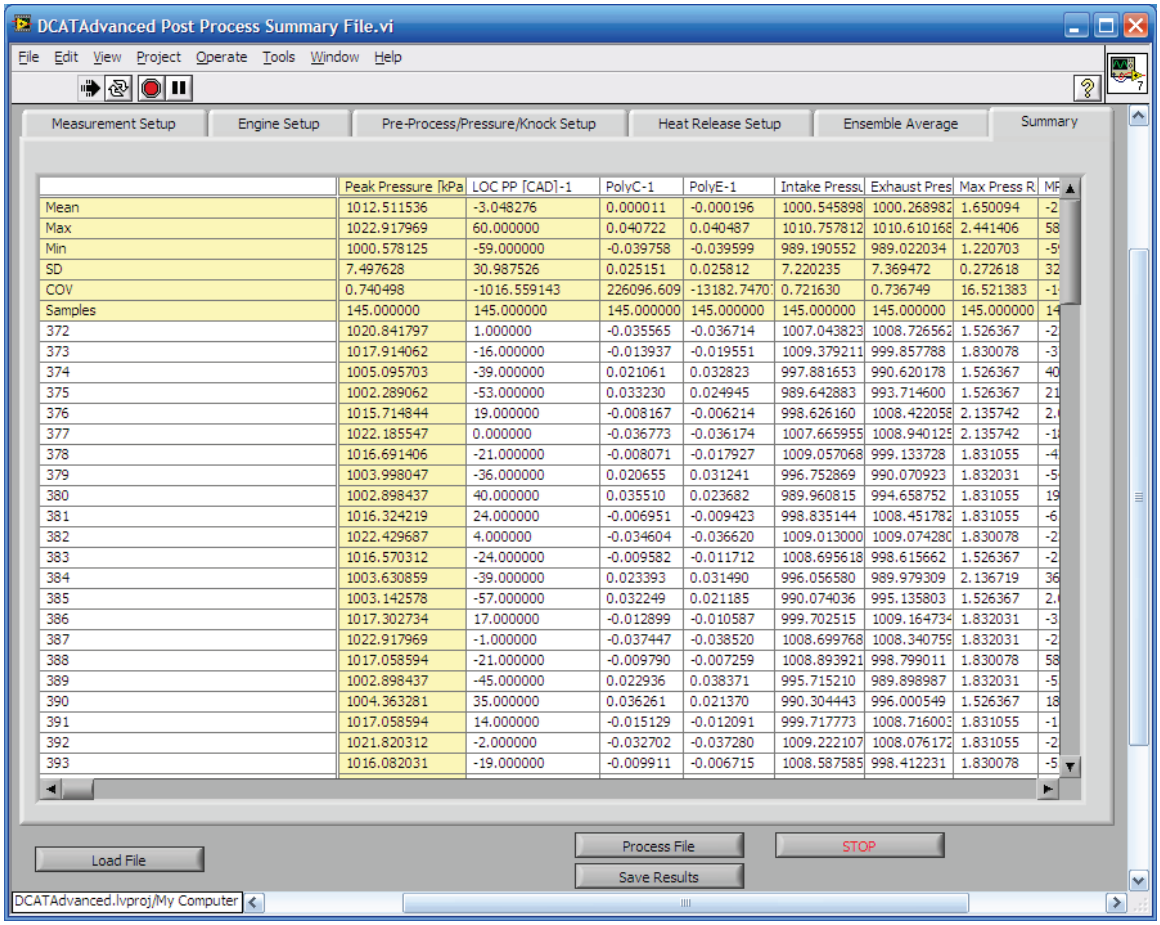
Summary_001 (root) / Cylinder_1 / Ensemble Average

Ready NUM

Figure 60: Summary File Import to Excel

Example Post Processing (Summary Files)

The summary post processing example is set up to display all of the configuration information and allow the user to easily insert their own post processing code. The front panel contains a tab control and several buttons to load and view files. The slider controls the cycle displayed in the graphs and tables. The button used for processing and saving do nothing in the example but are set up to allow the user to easily add processing and saving code for their specific needs. The example is not set up to allow the user to change the setup information because the properties have already been applied to all the calculated values.



	Peak Pressure [kPa	LOC PP [CAD]-1	PolyC-1	PolyE-1	Intake Pressu	Exhaust Pres	Max Press R	MF
Mean	1012.511536	-3.048276	0.000011	-0.000196	1000.545898	1000.268982	1.650094	-2
Max	1022.917969	60.000000	0.040722	0.040487	1010.757812	1010.610168	2.441406	58
Min	1000.578125	-59.000000	-0.039758	-0.039599	989.190552	989.022034	1.220703	-5
SD	7.497628	30.987526	0.025151	0.025812	7.220235	7.369472	0.272618	32
COV	0.740498	-1016.559143	226096.609	-13182.7470	0.721630	0.736749	16.521383	-1
Samples	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	145.000000	14
372	1020.841797	1.000000	-0.035565	-0.036714	1007.043823	1008.726562	1.526367	-2
373	1017.914062	-16.000000	-0.013937	-0.019551	1009.379211	999.857788	1.830078	-3
374	1005.095703	-39.000000	0.021061	0.032823	997.881653	990.620178	1.526367	40
375	1002.289062	-53.000000	0.033230	0.024945	989.642883	993.714600	1.526367	21
376	1015.714844	19.000000	-0.008167	-0.006214	998.626160	1008.422058	2.135742	2.
377	1022.185547	0.000000	-0.036773	-0.036174	1007.665955	1008.940125	2.135742	-1
378	1016.691406	-21.000000	-0.008071	-0.017927	1009.057068	999.133728	1.831055	-4
379	1003.998047	-36.000000	0.020655	0.031241	996.752869	990.070923	1.832031	-5
380	1002.898437	40.000000	0.035510	0.023682	989.960815	994.658752	1.831055	19
381	1016.324219	24.000000	-0.006951	-0.009423	998.835144	1008.451782	1.831055	-6
382	1022.429687	4.000000	-0.034604	-0.036620	1009.013000	1009.074280	1.830078	-2
383	1016.570312	-24.000000	-0.009582	-0.011712	1008.695618	998.615662	1.526367	-2
384	1003.630859	-39.000000	0.023393	0.031490	996.056580	989.979309	2.136719	36
385	1003.142578	-57.000000	0.032249	0.021185	990.074036	995.135803	1.526367	2.
386	1017.302734	17.000000	-0.012899	-0.010587	999.702515	1009.164734	1.832031	-3
387	1022.917969	-1.000000	-0.037447	-0.038520	1008.699768	1008.340759	1.832031	-2
388	1017.058594	-21.000000	-0.009790	-0.007259	1008.893921	998.799011	1.830078	58
389	1002.898437	-45.000000	0.022936	0.038371	995.715210	989.898987	1.832031	-5
390	1004.363281	35.000000	0.036261	0.021370	990.304443	996.000549	1.526367	18
391	1017.058594	14.000000	-0.015129	-0.012091	999.717773	1008.716003	1.831055	-1
392	1021.820312	-2.000000	-0.032702	-0.037280	1009.222107	1008.076172	1.831055	-2
393	1016.082031	-19.000000	-0.009911	-0.006715	1008.587585	998.412231	1.830078	-5

Figure 61: Summary Post Processor Example Front Panel

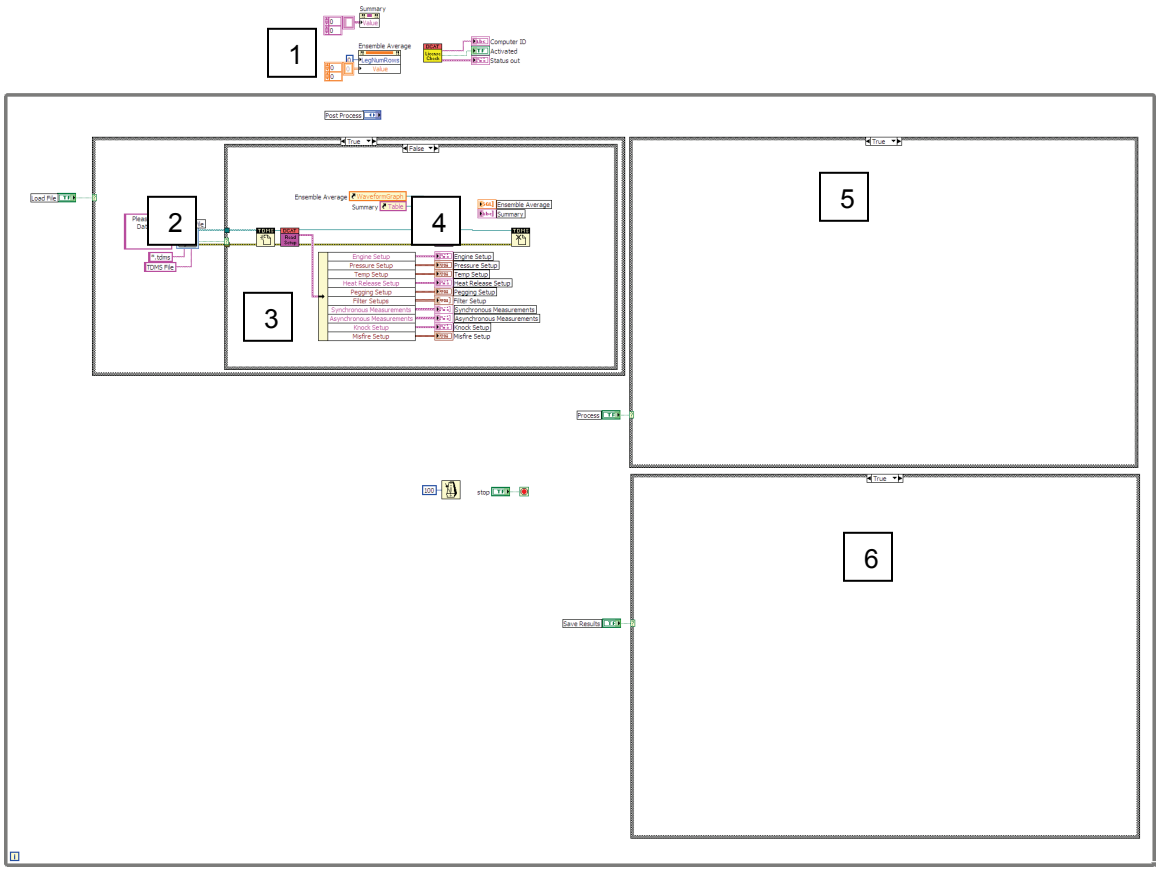


Figure 62: Summary Post Processing Example Block Diagram

1. Initialize the tables and plots. Check for a DCAT license.
2. Open the TDMS combustion data file.
3. Load the configuration data from the file.
4. Load the summary data from the file and populate the table and plot.
5. Section open for user defined processes.
6. Section open for user defined saving.

Opening a Raw Data File

A simple example of opening a data file has been included in the DCATAdvanced Example. Please note that the raw data files start at an arbitrary crank angle and must be aligned using data encoded in the file. The files store data as it is read from the device. Therefore, the individual cylinder offsets must also be accounted for. In Figure 63, cycle X is the first available cycle. LabVIEW VIs and examples have been provided to allow easy access to the cycle by cycle data without any need to know the specifics of the encoded data.

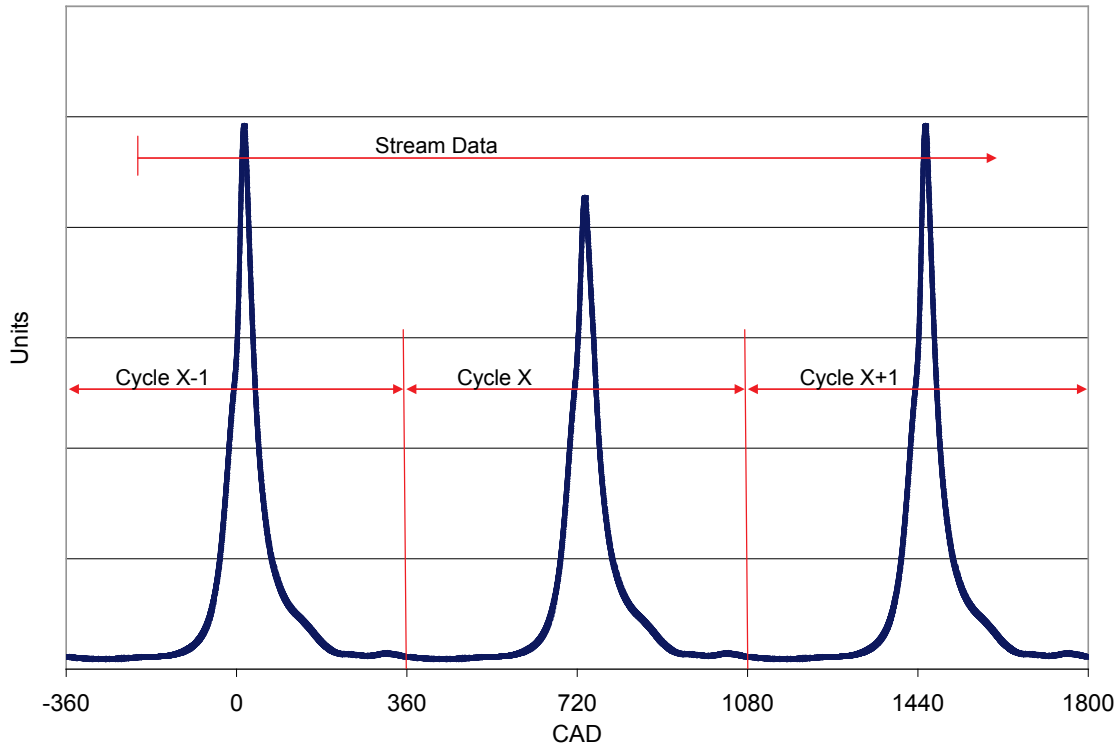


Figure 63: Raw Data File

The example opens the file and then allows the user to display a single cycle for a single cylinder. The plot automatically formats to show the measurement names originally saved with the data. Note that all the setup clusters are available and may be used for post processing or they may be modified before post processing.

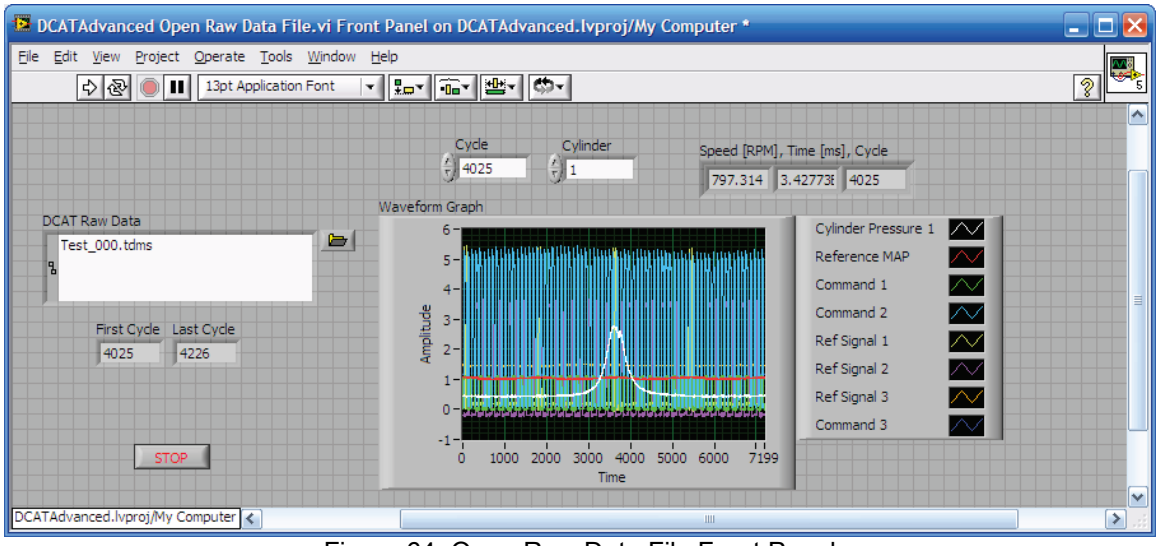


Figure 64: Open Raw Data File Front Panel

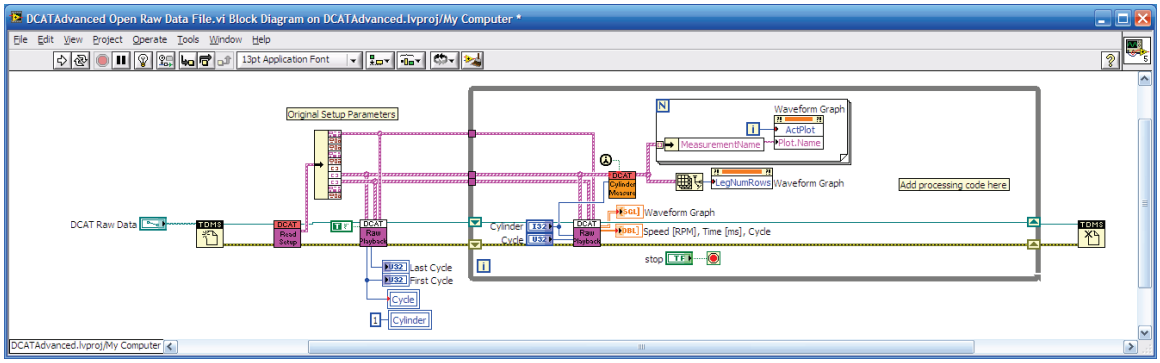


Figure 65: Open Raw Data File Block Diagram

The example can easily be modified to run all the cycles for all the cylinders in order to process an entire file. To automatically process the entire file, replace the while loop with two for loops that run for every cycle for every cylinder.

Example Post Processing (Raw Files)

The raw data post processing example shows and example of how to display and reprocess a streaming raw data file. The configuration settings can be changed and the statistics recalculated. The front panel contains a several tab controls and several other controls to load files, view data, and process the data.

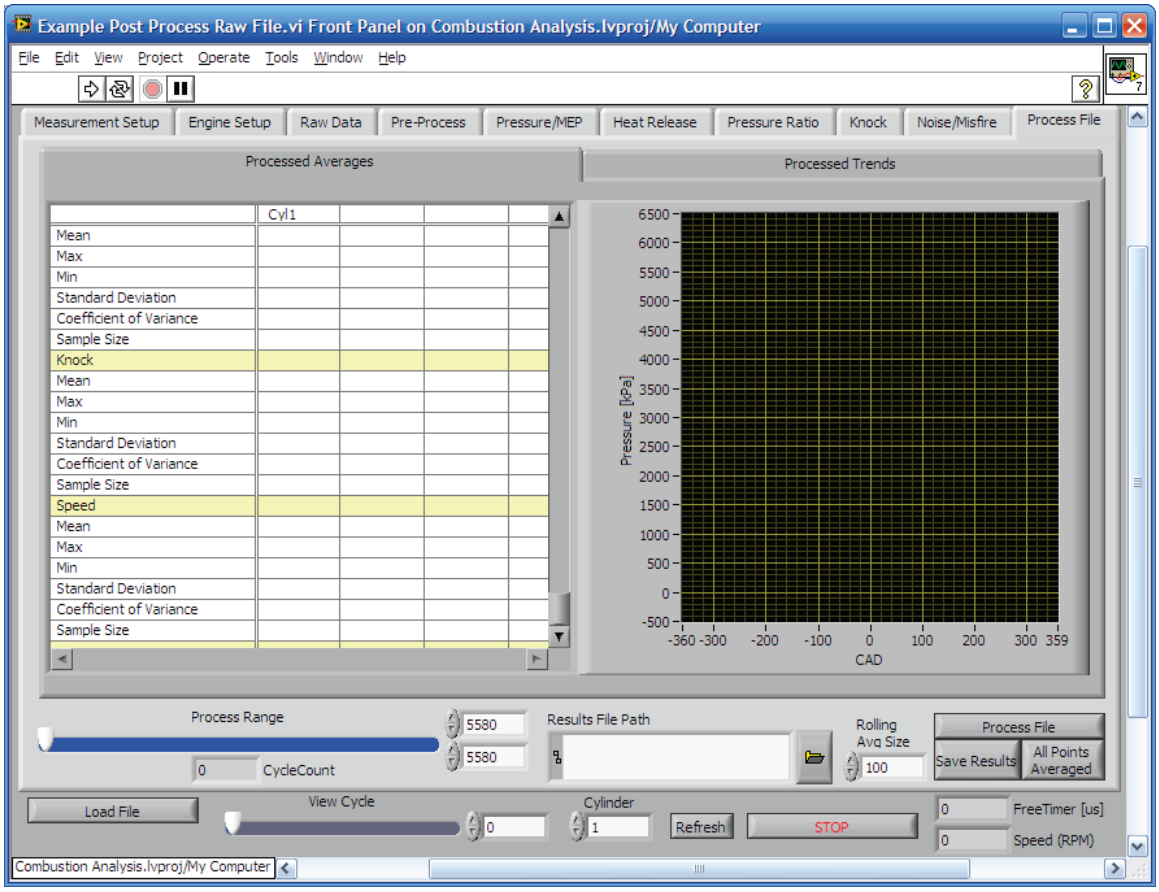


Figure 66: Raw Post Processing Example Front Panel

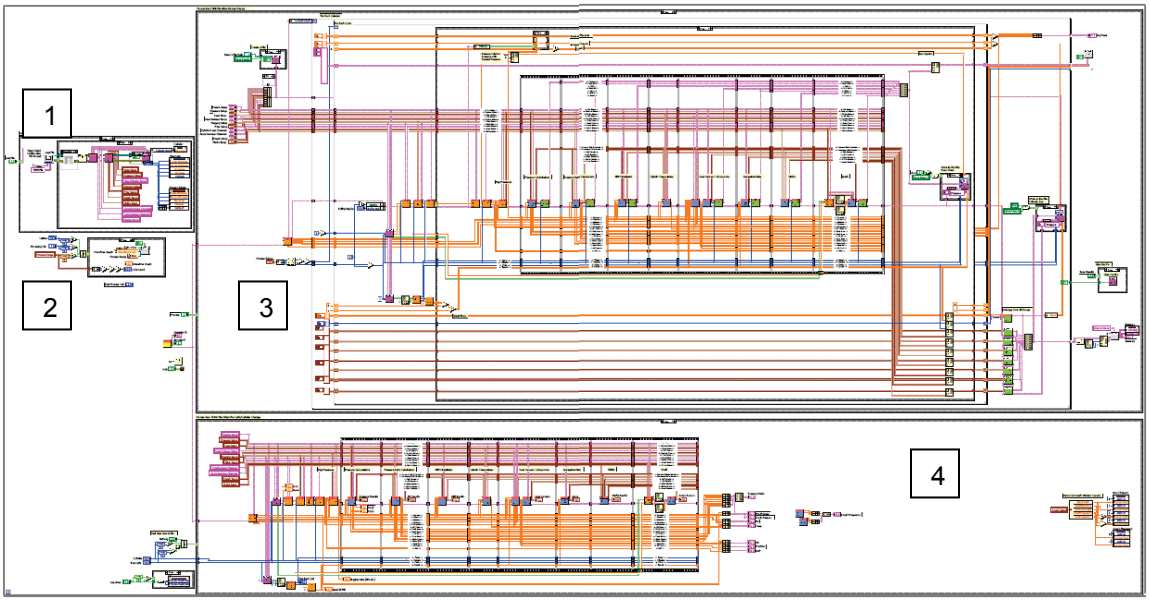


Figure 67: Raw Post Processing Example Block Diagram

1. Load the TDMS combustion file and all the configuration data. The file is indexed but the data is not loaded into LabVIEW. A single cycle of data is loaded when requested based on the indexes.
2. The cursor information on the trend data is updated when the process range slider moves.
3. Process all the values bracketed by the process range slider. The program processes all the cycles of one cylinder before moving to the next cylinder. The calculations are the same calculation used in the real time application but the statistics and display data is a little different. The results may be saved to a summary file.
4. Updates the preview of the data for the cycle indicated by the view cycle slider. The calculations are the same calculations used in the real time application but only run when the view cycle slider changes.

Case Studies

Several case studies were created to provide a general guide line for the setup and performance of the different systems. The case studies can be used as a general starting point to estimate the performance of a system.

Inline 4 Engine

This case study shows a basic setup of an inline 4 cylinder engine. The system needs to capture engine synchronous data for cylinder pressure sensors in each cylinder. The engine contains a ¼ deg crank shaft encoder. The system needs to calculate and display a 100 cycle rolling average of basic pressure metrics, mean effective pressures, simple heat release. In addition to the four in cylinder pressure sensors, a manifold pressure (used for pegging), injector current, fuel pressure, and exhaust pressure are to be displayed and recorded. No asynchronous sampling or next cycle control will be used.

To accomplish this, the recommended hardware will be used. Please note that the controller contains a memory upgrade.

	Type	NI Part	Description
PXI Chassis	Chassis	PXI-1031	4-Slot Chassis
Slot 1	Controller	PXI-8106 RT	2.16 GHz Dual Core (2GB Ram)
Slot 2	FPGA	PXI-7811R	1M Gate FPGA
Slot 3	S-Series	PXI-6123	8 x 500 kS/sec/ch Analog Inputs
Slot 4	-	-	-

As a starting point, the advanced DCAT example application will be used. To avoid losing any custom modifications when updating DCAT, the example application need to be copied new location.

The engine setup is as follows:

	Value	Units
Cylinder Count	4	
4-Stroke	Yes	
Bore	0.06	m
Stroke	0.06	m
Connection Rod Length	0.15	m
Pin Offset	0	m
Head Area	100	%
Crown Area	100	%
Compression Ratio	12.5	
Encoder Resolution	3600	ppr
Maximum Engine Speed	4500	RPM
TDC Offsets		
Cylinder TDC 1	0	CAD
Cylinder TDC 2	540	CAD
Cylinder TDC 3	360	CAD
Cylinder TDC 4	180	CAD

The calculations that are not needed may be disabled so that they are not using slowing the processors. For this case study, the following calculations were used:

Mean Effective Pressures	Yes
Misfire	Yes
Cylinder Temperature	No
Heat Release	Single Zone
Pressure Ratio	No
Knock	No
Noise	No

The asynchronous measurements should be set to disable and the synchronous measurement count should be set to 8. Please note that the manifold pressure is the only measurement associated with multiple cylinders. The injector current, fuel rail pressure, and manifold temperature are only associated with cylinder 1 so that they will only be aligned and displayed with cylinder 1. This was done to reduce the processor and memory load associated with aligning and displaying these signals. The manifold pressure is used for pegging of the cylinder pressure sensors and therefore should be associated with all applicable signals. The signals will only be recorded to the raw data file once regardless of the cylinder association. However, the ensemble average in the summary file will be calculated for each cylinder association. The measurement setups are as follows:

Measurement	Name	Filter	Cylinder	Scaling	Units	Channel
1	Cylinder Pressure 1	1	1	Gain	kPa	Dev1/ai0
2	Cylinder Pressure 2	1	2	Gain	kPa	Dev1/ai1
3	Cylinder Pressure 3	1	3	Gain	kPa	Dev1/ai2
4	Cylinder Pressure 4	1	4	Gain	kPa	Dev1/ai3
5	MAP	2	1,2,3,4	Gain + Offset	kPa	Dev1/ai4
6	Injector Current Probe	3	1	Gain + Offset	A	Dev1/ai5
7	Fuel Rail Pressure	4	1	Gain + Offset	MPa	Dev1/ai6
8	MAT	5	1	Gain + Offset	K	Dev1/ai7

The filters are grouped by the type of signal. All of the in cylinder pressure sensors will be filtered using the same method using filter #1. The filters are setup as follows:

	Use	Type	Frequency	Order
Filter 1	Cylinder Pressure	IIR Filter	15%	8
Filter 2	Manifold Pressure	IIR Filter	15%	8
Filter 3	Injector Current	None	-	-
Filter 4	Fuel Rail Pressure	None	-	-
Filter 5	Manifold Temperature	None	-	-

V8 Engine

The V8 engine case study demonstrates the use of multiple S-Series cards to collect synchronous and asynchronous data. The engine is equipped with a 1 deg resolution crank shaft encoder. Each cylinder requires a cylinder pressure sensor measurement. In addition, each bank needs its own manifold absolute pressure sensor for pegging. Ten total synchronous measurements are needed. Therefore, a second card must be used for sampling two of the channels. Please note that a 6122 could be used in place of one of the 6123s. The engine is also equipped with 4 knock sensors that will be recorded asynchronously at a rate of 100 kS/sec/ch. A 3M gate FPGA was selected to allow for additional FPGA functionality. An M-Series card is also selected to allow recording of slow speed signals. The code for the M-Series logging is not currently a part of the DCAT example applications but may be added by the user.

	Type	NI Part	Description
PXI Chassis	Chassis	PXI-1042	8-Slot Chassis
Slot 1	Controller	PXI-8106 RT	2.16 GHz Dual Core (2GB Ram)
Slot 2	FPGA	PXI-7813R	3M Gate FPGA
Slot 3	S-Series	PXI-6123	8 x 500 kS/sec/ch Analog Inputs
Slot 4	S-Series	PXI-6123	8 x 500 kS/sec/ch Analog Inputs
Slot 5	S-Series	PXI-6122	4 x 500 kS/sec/ch Analog Inputs
Slot 6	M-Series	PXI-6251	16 x 1 MS/sec Analog Inputs
Slot 7	-	-	-
Slot 8	-	-	-

As a starting point, the advanced DCAT example application will be used. To avoid losing any custom modifications when updating DCAT, the example application need to be copied new location.

The engine setup is as follows:

	Value	Units
Cylinder Count	8	
4-Stroke	Yes	
Bore	0.06	m
Stroke	0.06	m
Connection Rod Length	0.15	m
Pin Offset	0	m
Head Area	105	%
Crown Area	105	%
Compression Ratio	12.5	
Encoder Resolution	360	ppr
Maximum Engine Speed	3500	RPM
Asynchronous Sample Rate	100	kS/s
TDC Offsets		
Cylinder TDC 1	0	CAD
Cylinder TDC 2	270	CAD
Cylinder TDC 3	630	CAD
Cylinder TDC 4	540	CAD
Cylinder TDC 5	450	CAD
Cylinder TDC 6	360	CAD
Cylinder TDC 7	180	CAD
Cylinder TDC 8	90	CAD

The calculations that are not needed may be disabled so that they are not using slowing the processors. For this case study, the following calculations were used:

Mean Effective Pressures	Yes
Misfire	Yes
Cylinder Temperature	Yes
Heat Release	Single Zone
Pressure Ratio	No
Knock	Yes
Noise	No

The number of synchronous measurements was set to 10. The cylinder pressure measurements were all placed on the first card and the manifold pressures were placed on the second. Please note that the card is specified by the physical channel. The map signals are associated with the appropriate cylinders.

Measurement	Name	Filter	Cylinder	Scaling	Units	Channel
1	Cylinder Pressure 1	1	1	Gain	kPa	Dev1/ai0
2	Cylinder Pressure 2	1	2	Gain	kPa	Dev1/ai1
3	Cylinder Pressure 3	1	3	Gain	kPa	Dev1/ai2
4	Cylinder Pressure 4	1	4	Gain	kPa	Dev1/ai3
5	Cylinder Pressure 5	1	5	Gain	kPa	Dev1/ai4
6	Cylinder Pressure 6	1	6	Gain	kPa	Dev1/ai5
7	Cylinder Pressure 7	1	7	Gain	kPa	Dev1/ai6
8	Cylinder Pressure 8	1	8	Gain	kPa	Dev1/ai7
9	MAP Odd	2	1,3,5,7	Gain + Offset	kPa	Dev2/ai0
10	MAP Even	2	2,4,6,8	Gain + Offset	kPa	Dev2/ai1

In a similar way, the four asynchronous measurements are also configured.

Measurement	Name	Filter	Cylinder	Scaling	Units	Channel
1	Knock 1	0	1.3	Gain + Offset	V	Dev3/ai0
2	Knock 2	0	2,4	Gain + Offset	V	Dev3/ai1
3	Knock 3	0	5,7	Gain + Offset	V	Dev3/ai2
4	Knock 4	0	6,8	Gain + Offset	V	Dev3/ai3

The cylinder pressures are filtered using one filter definition and the manifold pressures are filtered using a second filter definition. The two filter definitions are setup to as follows:

	Use	Type	Frequency	Order
Filter 1	Cylinder Pressure	IIR Filter	15%	8
Filter 2	Manifold Pressure	IIR Filter	15%	8

M-Series or FPGA data logging is not currently part of the DCAT toolkit or example application. However, the user may choose to customize the example application to include logging of other data. Summary data may be added to the array of calculated values to include them in the file. Additionally, the summary or raw data files may be opened and written to using the standard TDMS. The FPGA timer and start trigger time should be saved and used to align data with the DCAT data.

Performance

The performance of the system depends on many factors including:

- PXI Controller Speed, Ram, & OS (Windows or Real Time)
- DAQ Hardware Limitations (Sampling Rate, Response Times)
- Engine Speed
- Cylinder/Measurement Count
- Asynchronous Sampling Rate
- Medium Speed Data (Number of Channels and Sample Rate)
- FPGA Data (Number of Channels and Sample Rate)
- Encoder Resolution
- Calculation Parameters
- HDD Throughput
- User Calculations

VI Benchmarking

Each of the following VIs was tested for the average execution required to run the VI. The VI Times are not intended to be used as actual execution time but as a relative execution. Larger number take longer to execute and smaller numbers execute faster. For example, the single zone heat release method calculations take approximately the same time as the mean effective pressure calculations.

VI Name	Test Name	VI Time [us]
Combustion Noise.vi	7200pts	622
Cylinder Temp.vi	7200pts	162
HR Analysis.vi	7200pts	223
HR Raw.vi	7200pts, Single Zone, Smoothing Factor = 1, -60 to 60	1181
HR Raw.vi	7200pts, Rassweiler Withrow, Smoothing Factor = 1, -60 to 60	3406
HR Raw.vi	7200pts, Single Zone Dual Transducer, Smoothing Factor = 1, -60 to 60	1015
HR Raw.vi	7200pts, Single Zone + Heat Transfer, PolyECor = Hayes, HeatTransferCor = Woschni 1978, Smoothing Factor = 1, -60 to 60	12531
HR Raw.vi	7200pts, Single Zone, Smoothing Factor = 1, -60 to 60 Boxcar 5	1359
Knock.vi	7200pts, Synchronous	766
Knock.vi	100kHz, 1000rpm, Asynchronous	1259
MEP.vi	7200pts	1039
Misfire.vi	100pts	1.074
Array Stats.vi	1000pts	68.8
Decode TDC.vi	7200pts	4.17
DeSerializeU16.vi		0.127
DeSerializeU32.vi		0.175
DeSerializeU8.vi		0.104
dx.vi	7200pts	0.013
Hold Init.vi		0.08
Resample Cycle.vi	7200pts No Change	6.8
Resample Cycle.vi	7200pts Even TDC Shift	877
Resample Cycle.vi	7200pts Resize to 720pts	66
Resize Array.vi	7200pts Resize To 720pts	0.011
Rolling Mean.vi	1000pts	0.12
Rolling Stats.vi	1000pts	17.5

DAQ Hardware

One of the key features limiting the maximum engine speed is the DAQ hardware. There are two cases where the DAQ hardware may limit the maximum engine speed. The first is the case where the DAQ equipment has a large sample rate but the encoder resolution is high. For example, a 500 kS/s/ch DAQ card is limited to ~8333RPM with a 3600 ppr encoder. The second case is where the DAQ hardware has a small sample rate. For example, a multiplexed sampling card at 1MS/s divided among 16 channels has a 62.5 kS/s/ch sampling rate.

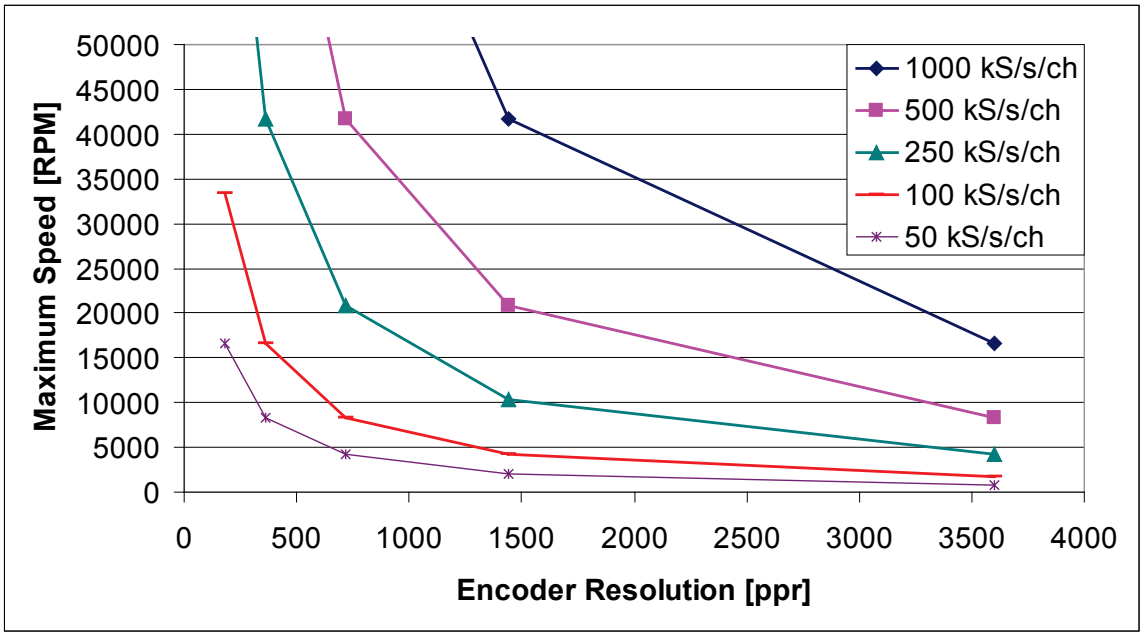


Figure 68: DAQ Hardware Performance

The data above was calculated using the following equation.

$$MS = \frac{60{,}000 * SR}{ER}$$

Where:

MS = Maximum Speed [RPM]
SR = Sample Rate [kS/s/ch]
ER = Encoder Resolution [ppr]

HDD

Streaming data to the disk requires that the speed of the hard drive is fast enough to keep up with the data waiting to write to it. Disk access is typically between 10MB/s and 70MB/s. However, due to seek times and writing header information, the actual speed at which the system can stream data to disk may be slower. The throughput of a specific HDD is dependant on many factors including the type of hard drive and the system that it is installed in. DCAT will buffer data in the ram if the disk is not fast enough to stream all the data to disk. However, the buffers have limited space and cannot sustain continuous streaming if the disk is not fast enough.

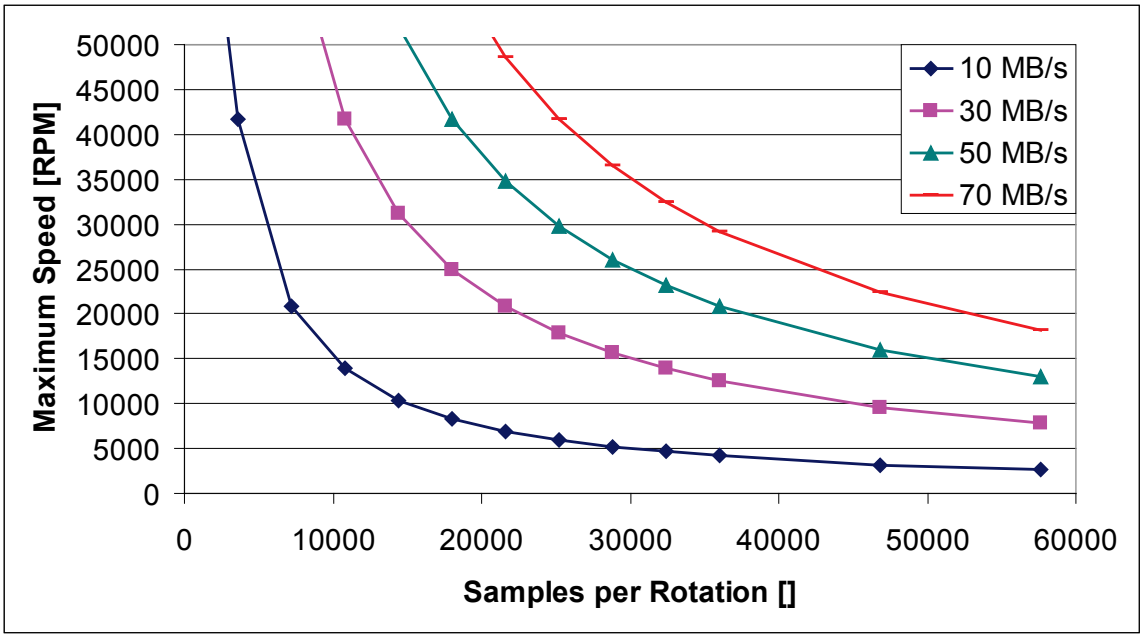


Figure 69: Hard Drive Performance

The data above was calculated using the following equation.

$$MS = \frac{60{,}000{,}000 * DS}{SR * 32/8}$$

Where:

MS = Maximum Speed [RPM]
SR = Sample per Rotation []
DS = Disk Speed [MB/s]

Frequently Asked Questions

How do I add my own calculations to DCAT?

- To add custom calculations, add them to the real time processing loop. There are a variety of different settings arrays and cluster. Please feel free to use existing setup information or calculated arrays where available (bore, stroke, mass fraction burned, etc.). If the calculations require a cycle to cycle history, make sure to read the cylinder number and account for multiple cylinders (DCAT VI are setup for up to 16 cylinders of data.)

How do I add a diesel injector current probe trace (or any other signal) to my display?

- To display and save any signal, simply add it to the measurement setup and associate it with the appropriate cylinder. It will be displayed on the data tab and saved in the raw data file. The MeasurementType should be set to undefined. The signal will be sampled engine-synchronously according to the encoder resolution.
- If it is necessary to perform calculation on the data or create a special display, the data can be separated from the data array using the “Get Cylinder Measurements” VI. The “Get Cylinder Measurements” VI returns an array of measurement setups corresponding to the data arrays. Searching for the desired measurement name will provide the index of the desired data array. Calculations and CalPoints can be added for the data.

When do I need to align the TDC offset?

- The TDC offset must be aligned any time the encoder position is modified. This includes changing the encoder.
- The TDC offset will change as a result of changing the encoder setup parameters. These parameters may have a small effect on the relation of TDC to the encoder Z pulse. Please see the section above for more details.
- The TDC offset will change as a result of changing the encoder resolution, even if the Z pulse remains constant.

How do I save the current configuration of DCAT?

- To save the current DCAT settings, use the CalVIEW “Save Current Calibration as Default” (purple) or the “Save Calibration As…” (blue) buttons on the CalVIEW interface. This saves a file to the LabVIEW default data directory. (This is typically under “.\My Documents\LabVIEW Data”) Calibration files have a .cdl extension.
- To load a previous DCAT calibration which was given a unique name and path, copy the desired calibration file into the LabVIEW default data directory and rename it to DCAT.cdl. Then start the DCAT target application. Please note that the calibration file is only read when the DCAT Target Program starts.

How do I add data to my summary data file?

- Properties (Once per File): Wire the data to the “User Property Values” and “User Property Names” inputs of the “Save Summary” VI. This information is stored as a text string in the final file. It can be used to hold setup information or large sets of information using the “Flatten to String” function.
- Summary Data (One Value per Cycle): Add calculated values to the “Stats Data” array. Please note that the parameters in the array must remain

constant while saving or the data file may be incorrect. Only the name and current value are saved in the file. The file statistics are calculated and save when the file closes.

- Ensemble Average Array: Add the data array and data name to the "Synchronous" and "Synchronous Data Names" array inputs of the "Save Summary" VI. The data will be ensemble averaged while saving a file.
- Other Data: Open a new TDMS reference using the "Current Path" output. Add your data using the TDMS write functions. Take care to not write data to the location used by DCAT. Close the TDMS reference when finished or the system may slow or crash.

How do I add data to my raw data file?

- Properties (Once per File): Wire the data to the "User Property Values" and "User Property Names" inputs of the "Stream Raw Data To Disk" VI. This information is stored as a text string in the final file. It can be used to hold setup information or large sets of information using the "Flatten to String" function.
- Other Data: Open a new TDMS reference using the "Current Path" output. Add your data using the TDMS write functions. Take care to not write data to the location used by DCAT. Close the TDMS reference when finished or the system may slow or crash.

Why does Excel fail to open a TDMS files?

- The TDMS Excel plug-in opens each property in a new column across the top of the spreadsheet. Property fields are used to store all (>255) of the configuration parameters. However, Excel 2003 is limited to 255 columns. Open the TDMS plug-in properties and turn off custom property imports.

How do I customize the user interface?

- The user interface is a standard LabVIEW VI. It may be modified with LabVIEW development tools to meet user requirements. The file should be saved in a new location so that updating DCAT installation does not overwrite the changes.

What font is used in the example applications?

- The example applications use Tahoma size 13pt as the default font. In some circumstances, a different size is used.

Why am I getting DAQ errors?

- Make sure that the hardware is set up properly in the Measurements and Automation Explorer and that the Measurement Setup tab has the proper physical channel names.
- Verify that the device is not being used for other DAQ tasks.
- Verify that a single DAQ card in not used for synchronous and asynchronous measurements.

Why is the top of the pressure signal clipped (flat)?

- The signals will appear flat when the charge amplifier is outputting a value greater than the DAQ card or the charge amplifier output is at its maximum.
- Reset the charge amplifier to set the zero offset pressure.
- Adjust the output gain of the charge amplifier.

What do I do if I am having trouble obtaining encoder sync?

- Verify that the Cam/Crank/TDC signals are set up correctly (polarity, offsets, extensions, etc.)
- The TDC signal and the crank signals should not have rising edges at the same time. This condition can cause a situation where the TDC signal and the crank signals toggle position and cause an error. (One signal may be read as true before the other.)
- Verify that the proper signals are connected to the FPGA. It may be helpful to pass the encoder signals through the FPGA and read them use an oscilloscope if available.
- The encoder synchronization is controlled by the engine tracking VIs. Please refer to the EPT documentation for details on the specific flags and inputs.

Why are the buffers or queues overflowing?

- The buffers fill when the computer system hardware is not able to keep up with the program demands. A combination of faster CPU, faster hard drive or additional RAM will alleviate the problem.
- Raw Queue: This buffer fills when data cannot be streamed to file as fast as the DAQ Loop rate. Try saving less data or allow the queue to empty before saving more data. Removing old data from a hard drive may also increase performance.
- RT Queue or NC Queue: These buffers fill when the combustion analysis calculations cannot be completed as fast as the DAQ Loop rate. Try disabling some of the calculations or reducing the encoder resolution.
- RT Align Buffer or NC Align Buffer: These buffers will fill when the combustion analysis calculations cannot be completed as fast as the combustion events. The calculation time must be less than (60 * N) / (RPM * Ncyl) seconds. (N=Strokes per Cycle, RPM = Engine Speed in RPM, Ncyl = Number of Cylinders) Ex. 4-storke @ 10000RPM and 16 cylinders = 750us. Try disabling some of the calculations or reducing the encoder resolution.
- The performance of the system depends on many variables including user defined functions. It may be necessary to remove some real time processing to achieve the desired performance.

Why are CalPoints behaving strangely?

- Verify that all the subVIs containing CalPoints (except DCAT VIs) are a part of the target in the project explorer.
- Stop the target application and rerun the CalVIEW Master setup.
- Please refer to the CalVIEW user's manual for more details.

Why are the timed loops not running?

- Change your timing source if you are running in Windows. The Windows OS will only work with the 1 kHz timing source.
- Check to see if the loop is receiving a stop signal. There are several error conditions that can stop the timed loops.

Why are the speed and the timer not working correctly?

- Check the PFI to P0 jumpers described in the hardware setup information. That data is transferred over these jumpers.

Why are the DCAT VIs running but are doing nothing.

- Please verify that you have a valid license key. Please see the section on installing and activating DCAT above.
- Verity that the “License DCAT” VI runs once before all other DCAT VIs and that it returns the “Activated” flag true.

Why do the timed loops behaving strangely?

- Timed loops must be configured to “Discard Missed Periods” and to not “Maintain Original Phase”.
- Timed loops should be set to a resetting timing source. This will reset the counter back to zero when DCAT switches to the running state.

Why does the real time controller crashed without warning.

- Reboot the controller and try running the application again.
- It may be possible to exceed the total physical memory of real time system. In this case, the real time controller will most likely crash (real time systems do not have virtual memory). The memory should be monitored using one of the tools available with LabVIEW. Remove some of the memory use or add more memory to the system to prevent the crashes. (Please check with National Instruments for details on upgrading memory in a real time system.)

Appendix A – Toolkit VIs

The toolkit VIs documentation in the following section may also be accessed in the context help in LabVIEW. To access the context help, press Ctrl-H in LabVIEW and place your cursor over the VI that you wish to see a description of.

Custom Control Clusters

Asynchronous Channels (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

- **Measurement Setup** (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Combustion Noise Results (Cluster, Combustion Noise Results.ctl) This control contains the results of the combustion noise calculations.

- **Noise [dB]** (Single Float) Displays the calculated combustion noise results.

Configuration Clusters (Cluster, Configuration Clusters.ctl) This control contains a cluster of all the setup clusters. It is used to pass all of the setup clusters to the saving VIs.

- **Engine Setup** (Cluster, Engine Setup.ctl) This control contains general engine setup information.
- **Pressure Setup** (Cluster, Pressure Setup.ctl) This control contains the basic pressure metrics calculations setup.
- **Temp Setup** (Cluster, Temp Setup.ctl) This control contains the cylinder temperature calculation setup.
- **Heat Release Setup** (Cluster, Heat Release Setup.ctl) This control contains the heat release calculations configuration.
- **Pegging Setup** (Cluster, Pegging Setup.ctl) This control contains the Pegging setup.

Filter Setups (Array, Filter Setups.ctl) This control contains the array of filter setup clusters. Each element in the array contains the setup information for a different filter.

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Asynchronous Measurements (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Knock Setup (Cluster, Knock Setup.ctl) This control contains the knock calculation setup.

Misfire Setup (Cluster, Misfire Setup.ctl) This control contains the misfire setup.

Medium Speed Setup (Cluster, Medium Speed Setup.ctl) This control contains the medium speed DAQ setup.

FPGA Data Setup (Array, FPGA Data Setup.ctl) This control contains the FPGA data stream setup.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

EngineName (String) Specifies the name of the test engine.

ProjectName (String) Specifies the name of the test project.

ClearanceVolume [liter] (Single Float) Specifies the cylinder volume at top dead center.

ConRodLength [m] (Single Float) Specifies the connection rod length.

CompressionRatio (Single Float) Specifies the cylinder compression ratio.

Stroke [m] (Single Float) Specifies the piston stroke.

Bore [m] (Single Float) Specifies the cylinder bore.

Stroke2-4 (Enum U32) Specifies a 2 or 4 stroke engine.

0: 4-Stroke: 2 engine revolution per cycle

1: 2-Stroke: 1 engine revolution per cycle

2: Other: User specified revolutions per cycle

AsyncRate [kS/s] (Single Float) Specifies the asynchronous sampling rate when applicable.

CVorCR (Enum U32) Specifies if the user enters clearance volume and compression ratio is calculated or if the user enters compression ratio and the clearance volume is calculated.

0: Clearance Volume: User enters clearance volume

1: Compression Ratio: User enters compression ratio

TDC_Offset [CAD] (Single Float) Specifies the absolute top dead center offset to apply to all signals. The value will automatically be adjusted to the nearest sample.

SampleResolution [ppr] (U32) Displays the final sampling resolution after the encoder extrapolation and encoder divide.

PinOffset [m] (Single Float) Specifies the offset of the pin connecting the connecting rod to the piston.

CrownArea [%] (Single Float) Specifies the piston crown surface area compared to the cylinder bore area.

HeadArea [%] (Single Float) Specifies the head surface area compared to the cylinder bore area.

DisplacementVolume [liter] (Single Float) Displays the piston displacement volume.

CylinderCount (U32) Specifies the number of cylinders connected to the system.

CylinderTDC [CAD] (Array) Specifies the absolute top dead center offsets for each cylinder. Firing order is determined from the top dead center offsets.

EncoderResolution [ppr] (U32) Specifies the actual resolution of the encoder.

EncoderExtrapEnable (Boolean) Specifies if the extrapolate the encoder is enabled. This can be used to increase the encoder resolution to a higher resolution using the engine position tracking capabilities.

EncoderDivide (U32) Used to reduce the sampling resolution. It is applied after the extrapolation. The default value is 1 which will leave the encoder at the extrapolated resolution if enabled. The final resulting value must be an positive, even integer.

EncoderExtrapLevel (U32) Specifies the level of extrapolation used by the engine position tracking. This value must match the FPGA engine position tracking VIs.

IVO [CAD] (Single Float) Specifies the nominal Intake Valve Opening.

IVC [CAD] (Single Float) Specifies the nominal Intake Valve Closing.

EVO [CAD] (Single Float) Specifies the nominal Exhaust Valve Opening

EVC [CAD] (Single Float) Specifies the nominal Exhaust Valve Closing.

MaxSpeed [RPM] (Single Float) Specifies the maximum engine speed. This value is used to determine the loop times and reduced resolution signals.

ReducedResolution [ppr] (U32) Displays the resolution of the encoder signal recorded with the asynchronous sampling.

ReducedDivide (U32) Displays the divide applied to the sampling resolution to obtain the reduced resolution.

Actual_TDC_Offset [CAD] (Single Float) Displays the actual top dead center offset used by the software. This value shows closest sample to the specified TDC_Offset.

TDC_Fine_Adj [CAD] (Single Float) Specifies a partial sample resolution adjustment of the measurement waveforms. The data will be resampled by a fractional amount in software and is therefore may slow the system.

EnsembleEnable (Boolean) Specifics if the ensemble averaging is enabled. The ensemble average of the cycle waveforms happens in the alignment VI.

EnsembleSize (U32) Specifies the number of cycles to ensemble average before data leaves the cylinder alignment.

Operator (String) Specifies the name of the engine operator.

NC_Cutoff [CAD] (Single Float) Specifies the crank angle past TDC to start the next cycle calculations. This value must be set so that the required calculations have enough data to calculate properly but should be as soon as possible to allow more time for calculations.

NC_Enable (Boolean) Specifies if the next cycle control calculations are enabled. The next cycle controls may be used to perform next cycle control based on the calculated parameters.

OtherStrokes (U32) Specifies the number of strokes when using other stroke.

VolumeFileEnable (Boolean) Specifies if the volume, differential volume, and area are read from a file instead of calculating the values based on the specified geometry.

VolumeFilePath (String) Specifies the volume file path when VolumeFileEnable is true.

WindowDataEnable (Boolean) Specifies if the cycle waveforms are windowed. When enabled, data outside the window will be set to zero.

WindowDataStart [CAD] (Single Float) Specifies the start of the data window.

WindowDataEnd [CAD] (Single Float) Specifies the end of the data window.

FPGA Data Setup (Array, FPGA Data Setup.ctl) This control contains the FPGA data stream setup.

FPGA FIFO Data Configuration (Cluster, FPGA Data Configuration.ctl) This control contains the FPGA Data configuration setup.

FPGA Measurement (Cluster, FPGA Data Configuration.ctl) This control contains the FPGA Data configuration setup.

FPGA_ID (U8) Specifies the ID number associated with the data stream. The ID is defined on in the FPGA to distinguish between the data streams.

FPGA_Name (String) Specifies the name of the FPGA measurement. This is used as the name of the channel saved in the summary data file.

FPGA_Units (String) Specifies the units of the measurement after the gain and offset are applied.

FPGA_Gain (Single Float) Specifies the gain required to convert the values from the FPGA to the specified units.

FPGA_Offset (Single Float) Specifies the offset required to convert the values from the FPGA to the specified units.

Filter Setup (Cluster, Filter Setup.ctl) This control contains the filter setup.

FilterMethod (Enum U32) Specifies the filtering method.

0: None: No filtering is applied to the measurements

1: Rolling N Cycles: Uses a rolling average method

2: Forward and Reverse IIR: Applies an IIR filter in forward and reverse directions to remove the phase shift

3: FIR Filter w/ Rollback: Applies a FIR filter and shifts the data by the phase shift

4: Boxcar Average: Averages each sample with the samples on each side of the current sample

Frequency [%] (Single Float) Specifies the filter cutoff frequency as a percent of the Nyquist frequency. It is used with the Forward and Reverse IIR filter and the FIR Filter w/ Rollback.

FilterN (I32) Specifies the number of cycles to use with the Rolling N Cycles method and the number of sample on each side of the current sample to average in the Boxcar Average method.

FilterOrder (I32) Specifies the order of the filter. It is used with the Forward and Reverse IIR filter and the FIR Filter w/ Rollback.

Filter Setups (Array, Filter Setups.ctl) This control contains the array of filter setup clusters. Each element in the array contains the setup information for a different filter.

Filter Setup (Cluster, Filter Setup.ctl) This control contains the filter setup.

Heat Release Results (Cluster, Heat Release Results.ctl) This control contains the heat release calculations results.

5%MFB [CAD] (Array) Displays the 5% mass fraction burned calculations results.

10%MFB [CAD] (Array) Displays the 10% mass fraction burned calculations results.

50%MFB [CAD] (Array) Displays the 50% mass fraction burned calculations results.

75%MFB [CAD] (Array) Displays the 75% mass fraction burned calculations results.

90%MFB [CAD] (Array) Displays the 90% mass fraction burned calculations results.

MHRR [J/CAD] (Array) Displays the maximum heat release rate calculations results.

LocMHRR [CAD] (Array) Displays the location of the maximum heat release rate calculations results.

Heat Release Setup (Cluster, Heat Release Setup.ctl) This control contains the heat release calculations configuration.

EndAngle [CAD] (Single Float) Specifies the end or window end of the heat release calculations.

StartAngle [CAD] (Single Float) Specifies the start or window start of the heat release calculations.

HeatReleaseMethod (Enum U32) Specifies the heat release method used to calculate heat release and mass fraction burned.

0: Single Zone: Single zone approximation without heat transfer.

1: Single Zone Dual Transducer: Single zone approximation using a main chamber and pre-chamber pressure sensor without heat transfer.

2: Single Zone + Heat Transfer: Single zone approximation with heat transfer.

3: Single Zone Dual Transducer + Heat Transfer: Single zone approximation using a main chamber and pre-chamber pressure sensor with heat transfer.

4: Modified Rassweiler and Withrow: Modified Rassweiler and Withrow approximation.

5: Pressure Ratio: Pressure ratio of measured pressure and calculated motoring pressure.

StartMethod (Enum U32) Specifies the method used to start the mass fraction burned calculations.

0: Constant: Calculations start at the specified start angle.

1: N Below Zero: Calculations start after the start angle when N consecutive samples are below zero.

2: Boxcar Average Below Zero: Calculations start after the start angle when the boxcar average is below zero.

3: N Above Zero: Calculations start after the start angle when N consecutive samples are above zero.

4: Boxcar Average Above Zero: Calculations start after the start angle when the boxcar average is above zero.

StartN (I32) Specifies N value used in some of the start of heat release calculations.

EndMethod (Enum U32) Specifies the method used to end the mass fraction burned calculations.

0: Constant: Calculations start at the specified start angle.

1: N Below Zero: Calculations start after the start angle when N consecutive samples are below zero.

2: Boxcar Average Below Zero: Calculations start after the start angle when the boxcar average is below zero.

EndN (I32) Specifies N value used in some of the end of heat release calculations.

T_Wall [K] (Single Float) Specifies the temperature of the cylinder wall.

HeatTranCoeff [W/m2-K] (Single Float) Specifies a constant heat transfer correlation.

Pre-ChamberVolume [liter] (Single Float) Specifies the pre-chamber volume used in the dual transducer heat release methods.

PolyECor (Heat Transfer) (Enum U32) Specifies the correlation for polytropic coefficient of expansion used in the heat transfer correlations.

0: Hayes (SAE860029): Correlation presented by Hayes

1: Indolene (SAE841359): Correlation for Indolene

2: Propane (SAE841359): Correlation for Propane

3: Custom (a*T(K)+b): User defined correlation

a (Single Float) Specifies the gain associated with the user defined polytropic coefficient correlation.

b (Single Float) Specifies the offset associated with the user defined polytropic coefficient correlation.

HeatTransferCor (Enum U32) Specifies the heat transfer correlation used with heat release methods containing heat transfer calculations.

0: Constant: User defined constant heat transfer correlation

1: Woschni 1978: Heat transfer correlation described by Woschni in 1978

2: Woschni 1990: Updated heat transfer correlation described by Woschni in 1990

3: Hohenberg: Heat transfer correlation described by Hohenberg

C1ge (Single Float) Specifies the Woschni correlation coefficient 1 for the gas exchange period. (Default = 6.18)

C1c (Single Float) Specifies the Woschni correlation coefficient 1 for the compression period. (Default = 2.28)

C1e (Single Float) Specifies the Woschni correlation coefficient 1 for the expansion period. (Default = 2.28)

C2ge (Single Float) Specifies the Woschni correlation coefficient 2 for the gas exchange period. (Default = 0)

C2c (Single Float) Specifies the Woschni correlation coefficient 2 for the compression period. (Default = 0)

C2e (Single Float) Specifies the Woschni correlation coefficient 2 for the expansion period. (Default = 0.00324)

TempProfile (Enum U32) Specifies the method used to determine the cylinder temperature profile. It is used with heat release methods containing heat transfer.

0: Constant: Wall temperature is assumed to be a constant value.

1: Linear: Wall temperature varies linearly from the top of the cylinder to the bottom of the cylinder.

T_BDC [K] (Single Float) Specifies the temperature of the cylinder wall at bottom dead center.

T_TDC [K] (Single Float) Specifies the temperature of the cylinder wall at top dead center.

SmoothingFactor [%] (Single Float) Specifies the gain associated with the user defined polytropic coefficient correlation.

Knock Results (Cluster, Knock Results.ctl) This control contains the knock calculations results.

Knock (Single Float) Displays the knock intensity value calculated using the BRIC method.

PressurePeakToPeak [kPa] (Single Float) Displays the peak to peak knock intensity value.

Knock Setup (Cluster, Knock Setup.ctl) This control contains the knock calculation setup.

LocationMethod (Enum U32) Specifies the method used to window the knock data.

0: Percentage of Cycle (Fast): Determines the window locations using a percentage of the cycle without consideration of the encoder signal. This method is faster but less accurate than the alternative methods.

1: Exact Crank Angle (Slow): Determines the window locations using the encoder signal. This method is slower but more accurate than the alternative methods.

StartKnock [CAD] (Single Float) Specifies the start of the knock window.

StartReference [CAD] (Single Float) Specifies the start of the reference window.

WindowLength [CAD] (Single Float) Specifies the length of the knock and reference windows.

LowCutoffFrequency [kHz] (Single Float) Specifies the lower cutoff frequency of the knock filter.

HighCutoffFrequency [kHz] (Single Float) Specifies the upper cutoff frequency of the knock filter.

Asynchronous (Boolean) Specifies the source of the knock signal.

ForwardCoefficients (Array) Specifies the forward filter coefficients. 1st dimension corresponds to each element in the Coefficient Sample Rate and Filter Structure arrays. The second dimension represents the set of coefficients.

ReverseCoefficients (Array) Specifies the reverse filter coefficients. 1st dimension corresponds to each element in the Coefficient Sample Rate and Filter Structure. The second dimension represents the set of coefficients.

CoefficientSampleRates [kHz] (Array) Specifies the sample rates of the forward and reverse coefficients. Each element in the array corresponds to a set of forward and reverse coefficients, filter structure. The element in the array matches the coefficients 1 dimensional element.

FilterStructure (Array) Specifies the filter structure of the forward and reverse coefficients. Each element in the array corresponds to a set of forward and reverse coefficients, filter structure. The element in the array matches the coefficients 1 dimensional element.

Threshold (Array) Specifies the knock intensity thresholds. One for each cylinder.

MinimumReferenceValue (Single Float) Specifies the minimum reference window value.

MEP Results (Cluster, MEP Results.ctl) This control contains the MEP calculations results.

- **IMEP [kPa]** (Array) Displays the gross Indicated Mean Effective Pressure.
- **PMEP [kPa]** (Single Float) Displays the Pumping Mean Effective Pressure.
- **NMEP [kPa]** (Single Float) Displays the Net Mean Effective Pressure.

MEPalt Results (Cluster, MEPalt Results.ctl) This control contains the alternative MEP calculations results.

- **IMEPalt [kPa]** (Single Float) Displays the alternative gross Indicated Mean Effective Pressure.
- **PMEPalt [kPa]** (Single Float) Displays the alternative Pumping Mean Effective Pressure.

MEParea Results (Cluster, MEParea Results.ctl) This control contains the area MEP calculations results.

- **IMEParea [kPa]** (Single Float) Displays the area gross Indicated Mean Effective Pressure.
- **PMEParea [kPa]** (Single Float) Displays the area Pumping Mean Effective Pressure.

Measurement (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

- **AssociatedCylinder** (U32) Specifies the cylinders associated with the measurement. Each bit in the number represents a cylinder starting with the least significant bit.
- **MeasurementName** (String) Specifies the name given to the measurement. This is used to store the channel in the output files.
- **Gain** (Single Float) Specifies the gain when using the "Gain Only" or "Gain and Offset" scaling methods.
- **Offset** (Single Float) Specifies the offset when using the "Gain and Offset" scaling method.

PhysicalChannel (Tag) Specifies the name of a DAQmx physical channel to use by the DCAT system. The DAQmx physical channel can be viewed and sometimes changed in the Measurements and Automation eXplorer. Example: "Dev1/ai7".

ScalingMethod (Enum U32) Specifies the method used for scaling the measurement data. The method determines which calibration inputs to use. The options are to use "Gain Only", "Gain and Offset", "Table", or "Polynomial". The table method is slow and the polynomial is very slow.

0: Gain Only: Uses only the gain to scale the measurement. Some alternative method for setting the offset may be used. Ex. Gain * x

1: Gain and Offset: Uses the gain and offset to scale the measurement. Ex. Gain * x + Offset

2: Table: Linearly interpolates values on the table to determine the output value. The output is bounded by the output array values on the table. The input array should be monotonically increasing.

3: Polynomial: Uses a polynomial to scale the measurement. This method should be avoided because the calculations are slow.

TableCalibration (Array) Specifies the table calibration to use when the Scaling Method is set to "Table". The table values are interpolated to determine the results.

Polynomial f(x) (String) Specifies the polynomial to use when scaling method is set to "polynomial". The polynomial is in the form of y=f(x). Example: "x*12+exp(x)".

MeasurementType (Enum U32) Specifies the type of measurement. This determines how the processing will use the channel.

0: Undefined: Specifies that the measurement is not one of the defined types.

1: Cylinder Pressure [kPa]: Specifies that the measurement is the primary cylinder pressure measurement. Only one measurement per cylinder should be defined as a cylinder pressure.

2: PreChamber Pressure [kPa]: Specifies that the measurement is a prechamber pressure. The prechamber should be accompanied by a main cylinder pressure for each cylinder.

3: Manifold Absolute Pressure [kPa]: Specifies that the measurement is a manifold absolute pressure. Only one measurement per cylinder should be defined as a manifold absolute pressure.

Filter (U8) Specifies the filter resource to be used with the measurement. The Filter details are specified under the filter setup.

Units (String) Specifies the name given to the measurement. This is used to store the channel in the output files.

Disable (Boolean) Specifies a disabled measurement. Stops the recording, processing, and saving of the measurement.

Medium Speed Setup (Cluster, Medium Speed Setup.ctl) This control contains the medium speed DAQ setup.

Error (Boolean) Displays an error in the medium speed setup.

Sample Rate (Single Float) Specifies the sample rate of the medium speed DAQ.

Measurement Names (Array) Specifies the names of the medium speed DAQ measurements.

Physical Channel (Array) Specifies the physical channels of the medium speed DAQ measurements.

Scaling (Array) Specifies the scaling type of the medium speed DAQ measurements.

Units (Array) Specifies the units of the medium speed DAQ measurements.

Gain (Array) Specifies the gain of the medium speed DAQ measurements.

Offset (Array) Specifies the offset of the medium speed DAQ measurements.

CJC Source (Array) Specifies the CJC sources of the thermocouple medium speed DAQ measurements.

CJC Value (Array) Specifies the CJC value of the thermocouple medium speed DAQ measurements.

Filter Cutoff (Array) Specifies the filter cutoff frequency of the medium speed DAQ measurements. The filter cutoff frequency must be less than the sample frequency. A value of zero turns off the filter.

Terminal Configuration (Array) Specifies the terminal configuration of the medium speed DAQ measurements.

Signal Max (Array) Specifies the signal maximum of the medium speed DAQ measurements. The value should be in volts for gain scaling and temperature for thermocouples.

Signal Min (Array) Specifies the signal minimum of the medium speed DAQ measurements. The value should be in volts for gain scaling and temperature for thermocouples.

Misfire Results (Cluster, Misfire Results.ctl) This control contains the results of the misfire calculations.

Misfire [] (Single Float) Displays the results of the misfire calculations. Misfire is represented by a value of 1.

Misfire Setup (Cluster, Misfire Setup.ctl) This control contains the misfire setup.

MisfireMethod (Enum U32) Specifies the method used to detect misfires.

0: Peak Pressure (% of Historic): Detects a misfire when the peak pressure is less than a specified percentage of the rolling average.

1: Peak Pressure (Absolute): Detects a misfire when the peak pressure is less than a specified value.

2: IMEP (% of Historic): Detects a misfire when the NMEP is less than a specified percentage of the rolling average.

3: IMEP (Absolute): Detects a misfire when the NMEP is less than a specified value.

PercentThreshold [%] (Single Float) Specifies the percent threshold when using a misfire detection method requiring a percent threshold.

PressureThreshold [kPa abs] (Single Float) Specifies the pressure threshold when using a misfire detection method requiring a pressure threshold.

HistoryLength (I32) Specifies a history length when using a misfire detection method that requires a history.

Pegging Setup (Cluster, Pegging Setup.ctl) This control contains the Pegging setup.

PegPressure [kPa] (Single Float) Specifies the pegging pressure.

PegLocation [CAD] (Single Float) Specifies the location of the center of pegging window.

PeggingMethod (Enum U32) Specifies the pegging pressure method.

0: None: No pegging is applied to the cylinder pressure trace.

1: Constant: Pegs the cylinder pressure trace to a user specified value at a user specified location.

2: Synchronous MAP: Pegs the cylinder pressure trace to the MAP sensor at a user specified location.

3: Polytropic: Pegs the cylinder pressure trace to a user defined PolyC value over the PolyC calculation window.

PegAverage [CAD] (Single Float) Specifies the length to window used in pegging the cylinder pressure.

PegPolyC Polytropic Coefficient (Single Float) Specifies the PolyC value used in polytropic pegging.

Pressure Ratio Results (Cluster, Pressure Ratio Results.ctl) This control contains the pressure ratio calculations results.

PR(10) (Single Float) Displays the pressure ratio at 10 CAD.

PR(25) (Single Float) Displays the pressure ratio at 25 CAD.

FPR (Single Float) Displays the final pressure ratio.

PRM10 (Single Float) Displays the pressure ratio management timing parameter at 10 CAD.

PRM25 (Single Float) Displays the pressure ratio management timing parameter at 25 CAD.

DILPAR (Single Float) Displays the pressure ratio management dilution parameter.

MPR (Single Float) Displays the modified pressure ratio.

PR(0) (Single Float) Displays the pressure ratio at 0 CAD.

Pressure Results (Cluster, Pressure Results.ctl) This control contains the basic pressure metrics calculations results.

PP [kPa abs] (Array) Displays the peak pressure calculations results.

LPP [CAD] (Array) Displays the location of peak pressure calculations results.

MPR [kPa/CAD] (Array) Displays the maximum pressure rise calculations results.

LMPR [CAD] (Array) Displays the location of maximum pressure rise calculations results.

PolyC (Array) Displays the polytropic coefficient of compression calculations results.

PolyE (Array) Displays the polytropic coefficient of expansion calculations results.

Intake [kPa abs] (Single Float) Displays the intake pressure based on cylinder pressure.

Exhaust [kPa abs] (Single Float) Displays the exhaust pressure based on cylinder pressure.

Pressure Setup (Cluster, Pressure Setup.ctl) This control contains the basic pressure metrics calculations setup.

PPMethod (Enum U32) Specifies the method used to calculate the pressure metrics calculations.

0: Standard: The peak pressure and related values are set using the maximum value in the window.

1: Spline (Very Slow): The peak pressure and related values are set using a spline curve fit near the maximum value.

StartPP [CAD] (Single Float) Specifies the start of the peak pressure calculation window.

EndPP [CAD] (Single Float) Specifies the end of the peak pressure calculation window.

StartExhaust [CAD] (Single Float) Specifies the start of the exhaust calculation window.

EndExhaust [CAD] (Single Float) Specifies the end of the exhaust calculation window.

StartIntake [CAD] (Single Float) Specifies the start of the intake calculation window.

EndIntake [CAD] (Single Float) Specifies the end of the intake calculation window.

StartPolyC [CAD] (Single Float) Specifies the start of the polytropic coefficient of compression calculation window.

EndPolyC [CAD] (Single Float) Specifies the end of the polytropic coefficient of compression calculation window.

StartPolyE [CAD] (Single Float) Specifies the start of the polytropic coefficient of expansion calculation window.

EndPolyE [CAD] (Single Float) Specifies the end of the polytropic coefficient of expansion calculation window.

Save Setup (Cluster, Save Setup.ctl) This control contains the file saving setup.

SaveMethod (Enum U32) Specifies the method used to end the file.

0: Cycles, Timed, Size: Stops the file when any of the desired conditions are met.

1: Cycles: Stops the file when the desired number of cycle has been collected.

2: Timed: Stops the file when the desired time has been collected.

3: Size: Stops the file when the desired file size is met.

MaxSize [MB] (Single Float) Specifies the maximum file size when using a save method that uses file size as a stopping condition.

FileName (String) Specifies the name of the file to save. The file name will automatically convert "XXX" to the current file number. The file name will also automatically convert the file name using the standard time formats.

MaxCycles (I32) Specifies the maximum file cycles when using a save method that uses file cycles as a stopping condition.

MaxTime [s] (Single Float) Specifies the maximum file time when using a save method that uses file time as a stopping condition.

Path (Path) Specifies the folder path were data is to be saved.

StartCounter (I32) Determines the file number when the reset counter turns true.

ResetCounter (Boolean) Resets the file count to the value of start counter when the value turns true.

Continuous (Boolean) Specifies the automatically start a new file when a file ends.

Comments (String) Specifies the user comments to be included in the file properties.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Name (String) Property Name.

Value (Single Float) Current Value.

Mean (Single Float) Mean: Mean Value.

Max (Single Float) Maximum Value.

Min (Single Float) Minimum Value.

Standard Deviation (Single Float) Standard Deviation.

Coefficient of Variance (Single Float) Coefficient of Variance = Standard Deviation / Mean

Sample Size (Single Float) Size of the sample used in the statistics.

Status (Cluster, Status.ctl) This control contains the status of the real time process.

error (Cluster) Contains error messages and error conditions.

Message Array (Array) Array of messages to be recorded in to the summary file.

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Temp Setup (Cluster, Temp Setup.ctl) This control contains the cylinder temperature calculation setup.

ExhaustTemp [K] (Single Float) Specifies the exhaust temperature at intake valve closing.

IntakePressure [kPa] (Single Float) Specifies the intake pressure at exhaust valve closing.

ExhaustPressure [kPa] (Single Float) Specifies the exhaust pressure at exhaust valve closing.

IntakeTemp [K] (Single Float) Specifies the intake temperature at intake valve closing.

ResidualFraction (Single Float) Specifies the residual gas fraction.

IVC_Window [CAD] (Single Float) Specifies the window, in degrees before intake valve closing, used to calculate the initial pressure.

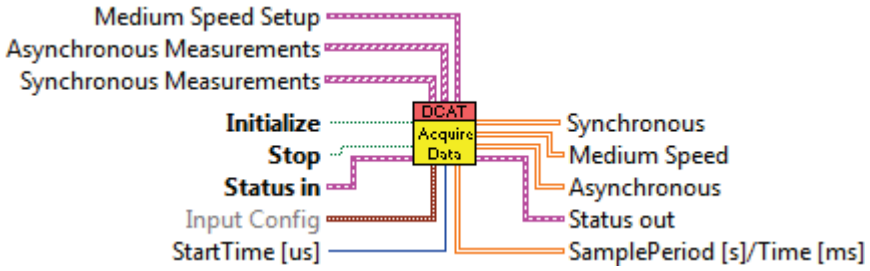
Wiebe Results (Cluster, Wiebe Results.ctl) This control contains the Wiebe function coefficients.

a (Double Float) Displays the a Wiebe coefficient.

m (Double Float) Displays the m Wiebe coefficient.

DAQ

Acquire Data (normal priority) This VI acquires the channel measurements from the hardware. The output is two sets of array containing the synchronous and asynchronous data.



Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous Measurements (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single

measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Input Config (Cluster, Input Config.ctl) This control contains the setup parameters for the synchronous and asynchronous DAQ tasks.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

Medium Speed Setup (Cluster, Medium Speed Setup.ctl) This control contains the medium speed DAQ setup.

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

StartTime [us] (U32) This control contains the DAQ start time from the FPGA. It is used to synchronize the medium speed DAQ and FPGA Data.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Stop (Boolean, Stop.ctl) This control stops the VI and clears the stored data in the VI. It should be used during the last loop of the cycle.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

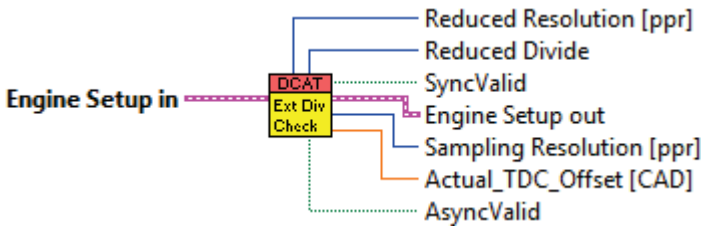
Synchronous Sample (Single Float) Synchronous Sample

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single

measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Effective Resolution (subroutine) This VI checks the Divide and Extrapolation values to determine if they are valid. If the values are not valid, the actual encoder resolution is used.



Actual TDC Offset [CAD] (Single Float) This control displays the actual top dead center offset that will be used by the data acquisition system. It is the closest integer sample offset to the desired top dead center offset.

Asynchronous Valid (Boolean) This control indicates true when the synchronous sampling setup is acceptable. It returns false when an error condition exists and overwrites the settings the default settings.

Engine Setup in (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Engine Setup out (Cluster, Engine Setup.ctl) This control contains general engine setup information.

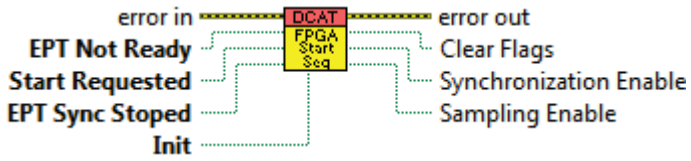
Reduced Divide (U32) This control displays the divide value required to achieve the reduced resolution.

Reduced Resolution [ppr] (U32) This control displays the reduced resolution encoder signal that is recorded with the asynchronous sampling to ensure that the encoder signal can be interpreted properly.

Sampling Resolution [ppr] (U32) This control displays the final sample resolution to be used by the data acquisition system.

Synchronous Valid (Boolean) This control indicates true when the asynchronous sampling setup is acceptable. It returns false when an error condition exists and overwrites the settings the default settings.

FPGA Start Seq (reentrant, normal priority) This VI automatically controls the start sequence for the EPT software in the FPGA. This VI will continually try to clear errors and establish synchronization. After synchronization, the VI will not try to reestablish synchronization if it is lost. These controls may be replaced by manual controls if desired.



Clear Flags (Boolean) This control should be wired to clear all engine position tracking flags.

EPT Not Ready (Boolean) This control should be set true when the engine position tracking is reporting some flag that prevents it from running. For the encoder pattern, this control is a logical or of the missed cam flag, missed crank flag, and the crank stalled.

EPT Sync Stopped (Boolean) This control should be connected to the engine position tracking synchronization stopped flag.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Sampling Enable (Boolean) This control should be wired to the sampling enable for the DCAT sampling supervisor VI in the FPGA.

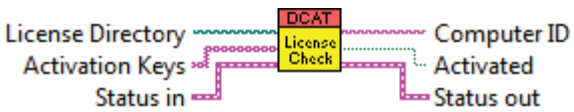
Start Requested (Boolean) This control should be set true and held true in order when sampling. Setting the control to false will reset the VI.

Synchronization Enable (Boolean) This control should be connected to the engine position tracking synchronization enable control.

error in (Cluster) This control is the standard LabVIEW error cluster.

error out (Cluster) This control is the standard LabVIEW error cluster.

License DCAT (normal priority) This VI checks the Activation Key and unlocks the DCAT SubVIs. This VI must be the first VI on the Status Control Path. An Activation Key may be provided on the block diagram. If no key is provided, the license.dat file in the license directory will be used. The default license directory is "C:\ni-rt\System\".



Activated (Boolean) This control indicates if the system is activated or un-activated.

Activation Keys (Array) The activation keys may be used as an alternative to using a file containing the activation keys. If this control contains any values, the licensing file is ignored.

Activation Key (String) Activation Key

Computer ID (String) The computer ID is a unique number generated based on the hardware of your system. There is not enough information encoded in the number to reconstruct any specific information about the system. The computer ID is used to generate activation keys.

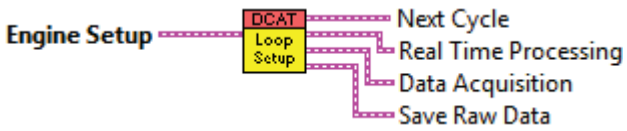
License Directory (Path) This control specifies the directory where the "license.dat" file is located. The path should always be a local path to the directory. The default path is "c:\ni-rt\system".

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Loop Setup (reentrant, normal priority) This VI takes the engine setup and automatically configures the setup for the timed loops. The loops should be set to balance the overhead associated with the and quick response times. The loops are set to run on the 1MHz Resetting clock when available and the 1kHz Resetting clock otherwise. The timed loops may be controlled by the user if desired.

The real time processing loop is set by the following calculations:
(60 * N) / (RPM * Ncyl) = Loop Time in sec
N=Strokes per Cycle
RPM = Engine Speed in RPM
Ncyl = Number of Cylinders
Example: 4-storke @ 10000RPM and 16 cylinders = 750us



Data Acquisition (Cluster, Loop Setup.ctl) This control contains loop setup information.

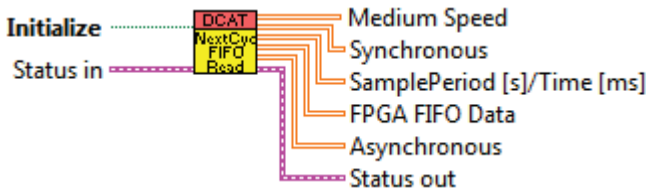
Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Next Cycle (Cluster, Loop Setup.ctl) This control contains loop setup information.

Real Time Processing (Cluster, Loop Setup.ctl) This control contains loop setup information.

Save Raw Data (Cluster, Loop Setup.ctl) This control contains loop setup information.

Next Cycle FIFO Read (normal priority) This VI reads the queue of raw data for the next cycle control.



Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

FPGA FIFO Data (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

FPGA Data (Double Float) FPGA Data

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the

current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

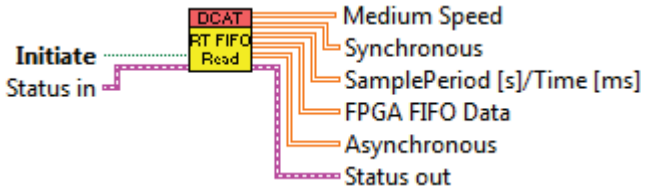
Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

RT FIFO Read (normal priority) This VI reads the queue of raw data for the real time processing VIs.



Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

FPGA FIFO Data (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

FPGA Data (Double Float) FPGA Data

Initiate (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the

timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

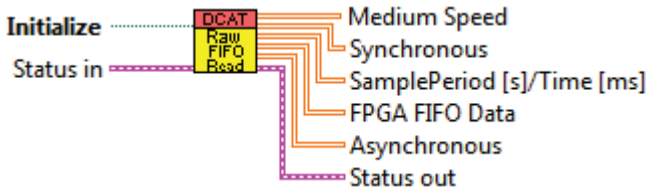
Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Stream Raw FIFO Read (normal priority) This VI reads the queue of raw data for the streaming to disk VIs.



Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

FPGA FIFO Data (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

FPGA Data (Double Float) FPGA Data

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

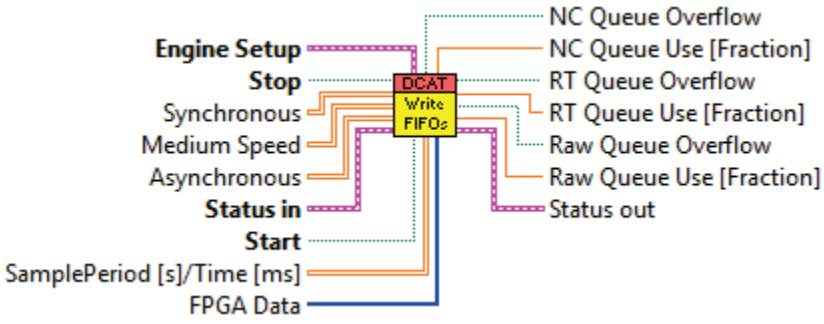
Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Write All FIFOs (normal priority) This VI combines the Queue writing for the next cycle loop, real time calculations loop, and the raw file saving loop. On systems where memory is limited, this VI can be replaced with the individual VIs for only the queues that are needed.



Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

FPGA Data (Array) This control contains the FPGA data directly from the FIFO.

FPGA Data (U64) FPGA Data

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

NC Queue Overflow (Boolean, Queue Overflow.ctl) This control displays true if any of the queues are full.

NC Queue Use [Fraction] (Single Float, Buffer Use.ctl) This control displays the fraction of the buffer or queue that is currently used.

RT Queue Overflow (Boolean, Queue Overflow.ctl) This control displays true if any of the queues are full.

RT Queue Use [Fraction] (Single Float, Buffer Use.ctl) This control displays the fraction of the buffer or queue that is currently used.

Raw Queue Overflow (Boolean, Queue Overflow.ctl) This control displays true if any of the queues are full.

Raw Queue Use [Fraction] (Single Float, Buffer Use.ctl) This control displays the fraction of the buffer or queue that is currently used.

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

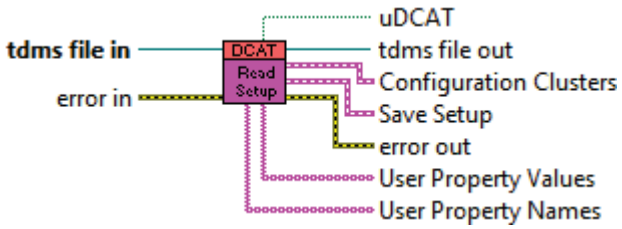
Stop (Boolean, Stop.ctl) This control stops the VI and clears the stored data in the VI. It should be used during the last loop of the cycle.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

File

Read Properties from TDMS (normal priority) This VI reads the channel setups, file saving setup, and engine setup from the TDMS file properties.



Configuration Clusters (Cluster, Configuration Clusters.ctl) This control contains a cluster of all the setup clusters. It is used to pass all of the setup clusters to the saving VIs.

Save Setup (Cluster, Save Setup.ctl) This control contains the file saving setup.

TDMS file in (Refnum) This control is a standard TDMS file reference.

TDMS file out (Refnum) This control is a standard TDMS file reference.

User Property Names (Array, User Property Names.ctl) This is an array of user property names with each element corresponding to each element in the user property values array.

User Property Name (String) User Property Name

User Property Values (Array, User Property Values.ctl) This is an array of user property values with each element corresponding to each element in the

user property names array. The "flatten to string" VI may be used to convert any data type to a string.

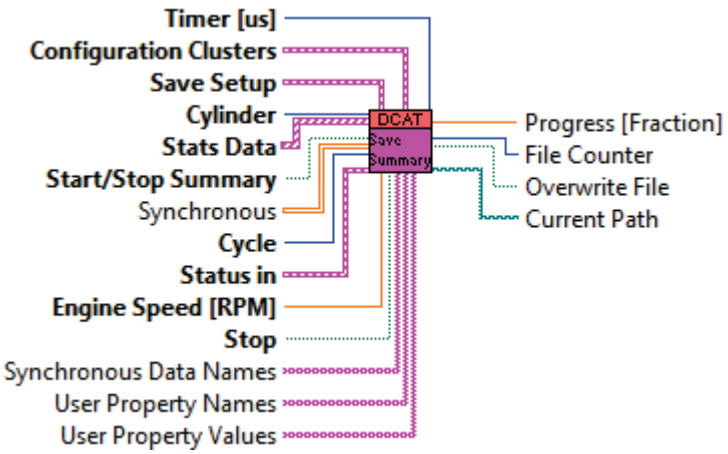
User Property Value (String) User Property Value

error in (Cluster) This control is a standard LabVIEW error cluster.

error out (Cluster) This control is a standard LabVIEW error cluster.

uDCAT (Boolean, uDCAT.ctl) This control turns on the uDCAT setup. It only needs to be connected when using uDCAT.

Save Summary (normal priority) This VI saves the results of the stats calculations to a TDMS file. The TDMS group is the cylinder number and measurement name. The TDMS channel is the calculated parameter. The error cluster and cycle number are also recorded. When the specified time or file size requirement is met, a new file is created. The engine setup and save setup are saved in the file. This VI is controlled by the "Control Saving-Init" VI through the status cluster.



User Property Names (Array, User Property Names.ctl) This is an array of user property names with each element corresponding to each element in the user property values array.

User Property Name (String) User Property Name

Configuration Clusters (Cluster, Configuration Clusters.ctl) This control contains a cluster of all the setup clusters. It is used to pass all of the setup clusters to the saving VIs.

Current Path (Path) This control displays the current file path. A path is only displayed when a file is in the process of saving.

Cycle (U32, Cycle.ctl) Cylinder associated with the current data set. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

File Counter (I32) Displays the current position of the file counter. This number is used in the name of the next file if desired.

Overwrite File (Boolean) This control displays true if the current settings for the file name and path already contain a file. This is the only warning that a file will be overwritten.

Progress [Fraction] (Single Float) This control displays the current file progress as a fraction of the specified file size.

Save Setup (Cluster, Save Setup.ctl) This control contains the file saving setup.

Start/Stop Summary (Boolean) This control starts or stops the summary file saving on the rising edge.

Stats Data (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

- **Stats** (Cluster, Stats.ctl) This control contains calculated statistical results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Stop (Boolean, Stop.ctl) This control stops the VI and clears the stored data in the VI. It should be used during the last loop of the cycle.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Synchronous Sample

Synchronous Data Names (Array) This array control contains a name for each synchronous data set in the synchronous array. The names are used as the channel name in the TDMS file and should be unique.

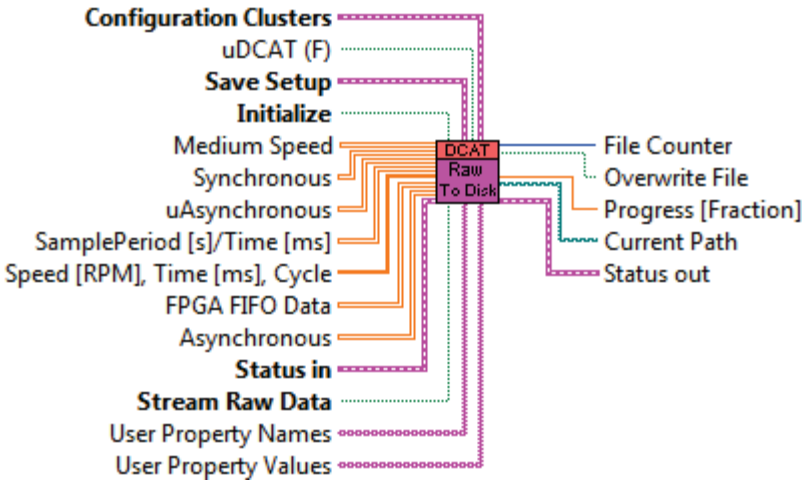
- **Data Name** (String) Synchronous Data Set Name

Timer [us] (U32, Timer.ctl) Timer contains the current value of the FPGA free timer. The time is updated once every cycle.

User Property Values (Array, User Property Values.ctl) This is an array of user property values with each element corresponding to each element in the user property names array. The "flatten to string" VI may be used to convert any data type to a string.

User Property Value (String) User Property Value

Stream Raw Data To Disk (normal priority) This VI streams the raw data to disk. Two groups are created, one for the synchronous data and one for the asynchronous data. When the save time, file size, or cycle count requirement is met, a new file is created. The engine setup, save setup, and channel setups are saved as properties in the file.



User Property Names (Array, User Property Names.ctl) This is an array of user property names with each element corresponding to each element in the user property values array.

User Property Name (String) User Property Name

Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Configuration Clusters (Cluster, Configuration Clusters.ctl) This control contains a cluster of all the setup clusters. It is used to pass all of the setup clusters to the saving VIs.

Current Path (Path) This control displays the current file path. A path is only displayed when a file is in the process of saving.

FPGA FIFO Data (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

- **FPGA Data** (Double Float) FPGA Data

File Counter (I32) Displays the current position of the file counter. This number is used in the name of the next file if desired.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

- **Medium Speed Sample** (Double Float) Medium Speed Sample

Overwrite File (Boolean) This control displays true if the current settings for the file name and path already contain a file. This is the only warning that a file will be overwritten.

Progress [Fraction] (Single Float) This control displays the current file progress as a fraction of the specified file size.

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

- **SamplePerid [s] / Time [ms]** (Double Float) Sample Period [s] / Time [ms]

Save Setup (Cluster, Save Setup.ctl) This control contains the file saving setup.

Speed [RPM], Time [ms], Cycle (Array, STC.ctl) This array control contains the speed, time, and cycle. The 0 index in the array is always speed. The 1 index is always time. The 2 index is always cycle number. This array is only used by uDCAT.

- **Value** (Double Float) Speed, Time, or Cycle

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Stream Raw Data (Boolean) This control starts or stops the raw data streaming to disk on a rising edge.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

User Property Values (Array, User Property Values.ctl) This is an array of user property values with each element corresponding to each element in the user property names array. The flatten to string VI may be used to convert any data type to a string.

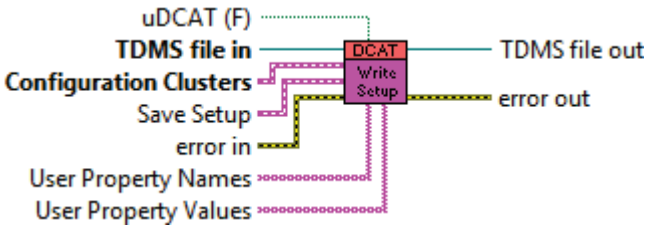
User Property Value (String) User Property Value

uAsynchronous (Array, uAsynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Double Float) Asynchronous Sample

uDCAT (F) (Boolean, uDCAT.ctl) This control turns on the uDCAT setup. It only needs to be connected when using uDCAT.

Write Properties to TDMS (reentrant, normal priority) This VI writes the pressure setup, heat release setup, filter setup, pegging setup, knock setup, misfire setup, and temperature setup to the TDMS file properties.



User Property Names (Array, User Property Names.ctl) This is an array of user property names with each element corresponding to each element in the user property values array.

User Property Name (String) User Property Name

Configuration Clusters (Cluster, Configuration Clusters.ctl) This control contains a cluster of all the setup clusters. It is used to pass all of the setup clusters to the saving VIs.

Save Setup (Cluster, Save Setup.ctl) This control contains the file saving setup.

TDMS file in (Refnum)

TDMS file out (Refnum)

User Property Values (Array, User Property Values.ctl) This is an array of user property values with each element corresponding to each element in the user property names array. The flatten to string VI may be used to convert any data type to a string.

User Property Value (String) User Property Value

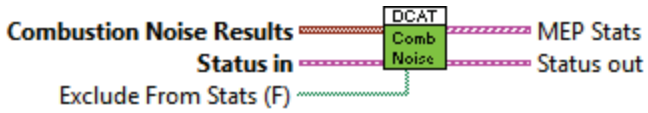
error in (Cluster) This control is a standard LabVIEW error cluster.

error out (Cluster) This control is a standard LabVIEW error cluster.

uDCAT (F) (Boolean, uDCAT.ctl) This control turns on the uDCAT setup. It only needs to be connected when using uDCAT.

PostProcess

Combustion Noise Stats Post Process (normal priority) This VI calculates the statistics for an array of the calculated combustion noise analysis values.



Combustion Noise Results (Array) This array control contains combustion noise results.

Combustion Noise Results (Cluster, Combustion Noise Results.ctl) Contains the results of the combustion noise results.

Exclude From Stats (F) (Array, Exclude From Stats Array.ctl) This control contains an array that can be used to exclude individual values from the statistics calculations. If the array is empty, all values will be used.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

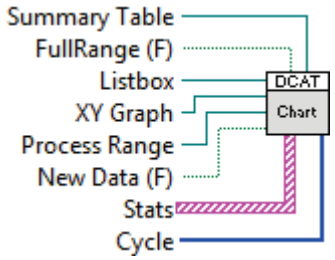
Noise Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Control Chart PP (normal priority) This VI controls a waveform graph (2d single) with 2 cursors to show a processed trend. This VI updates the data when New Data is true. Data is only needed when New Data is true. When New Data is false, the VI updates a trend graph of all cylinders for the parameter specified by the selected item in the list box. The process range slider defines the position of the cursors. Only the values that were processed will show up in the plot, therefore, it may be desirable to first run the processing 1 time with the entire range selected.



Cycle (Array) This control contains the cycle number for each cycle of processed data.

Cycle (U32) Cycle Number

Full Range (F) (Boolean) When true, this control sets the data processing range to the maximum values.

Listbox (Refnum) This control is a listbox reference. The listbox is used select the parameters shown on the XY graph.

New Data (F) (Boolean) This control should be set true when new data is available. It will replace the stored data when new data is available.

Process Range (Refnum) This control is a slider control reference. The slider should have two cursors. It is used to select the range of data for processing.

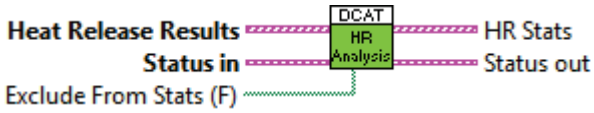
Stats (Array) This control contains all the statistical data for all cylinder, all cycles, and all properties.

Stats Out (Cluster, Stats.ctl) This control contains calculated statistical results.

Summary Table (Refnum)

XY Graph (Refnum) This control is a reference to an XY graph. The XY graph is used to display the trends of the properties.

HR Stats Post Process (normal priority) This VI calculates the statistics for an array of the calculated heat release analysis values.



Exclude From Stats (F) (Array, Exclude From Stats Array.ctl) This control contains an array that can be used to exclude individual values from the statistics calculations. If the array is empty, all values will be used.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Heat Release Results (Array) This array control contains heat release results.

Heat Release Results (Cluster, Heat Release Results.ctl) This array control contains combustion noise results.

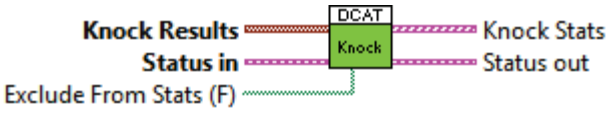
Heat Release Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Knock Stats Post Process (reentrant, normal priority) This VI calculates the statistics for an array of the calculated knock analysis values.



Exclude From Stats (F) (Array, Exclude From Stats Array.ctl) This control contains an array that can be used to exclude individual values from the statistics calculations. If the array is empty, all values will be used.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Knock Results (Array) This array control contains knock results.

Knock Results (Cluster, Knock Results.ctl) This control contains the knock calculations results.

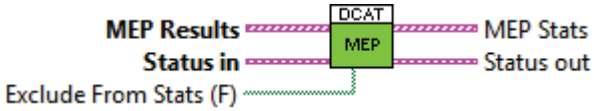
Knock Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

MEP Stats Post Process (normal priority) This VI calculates the statistics for an array of the calculated mean effective pressure analysis values.



Exclude From Stats (F) (Array, Exclude From Stats Array.ctl) This control contains an array that can be used to exclude individual values from the statistics calculations. If the array is empty, all values will be used.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to

determine the average peak pressure for all cylinder events that are not considered a misfire.

MEP Results (Array) This array control contains MEP results.

MEP Results (Cluster, MEP Results.ctl) This control contains the MEP calculations results.

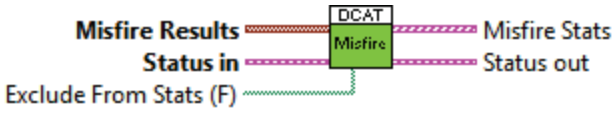
MEP Stats (Array, Statistics Array.ctl) This array control contains mean effective pressure results.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Misfire Stats Post Process (reentrant, normal priority) This VI calculates the statistics for an array of the calculated misfire analysis values.



Exclude From Stats (F) (Array, Exclude From Stats Array.ctl) This control contains an array that can be used to exclude individual values from the statistics calculations. If the array is empty, all values will be used.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Misfire Results (Array) This array control contains misfire results.

Misfire Results (Cluster, Misfire Results.ctl) This control contains the results of the misfire calculations.

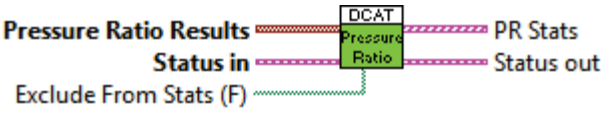
Misfire Stats (Array, Statistics Array.ctl) This array control contains misfire results.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Pressure Ratio Stats Post Process (normal priority) This VI calculates the statistics for an array of the calculated pressure ratio analysis values.



Exclude From Stats (F) (Array, Exclude From Stats Array.ctl) This control contains an array that can be used to exclude individual values from the statistics calculations. If the array is empty, all values will be used.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Pressure Ratio Results (Array) This array control contains pressure ratio results.

Pressure Ratio Results (Cluster, Pressure Ratio Results.ctl) This control contains the pressure ratio calculations results.

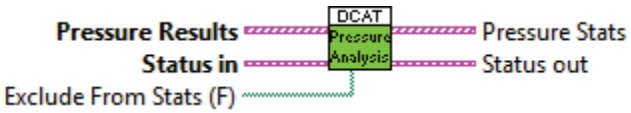
Pressure Ratio Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Pressure Stats Post Process (normal priority) This VI calculates the statistics for an array of the calculated pressure metrics analysis values.



Exclude From Stats (F) (Array, Exclude From Stats Array.ctl) This control contains an array that can be used to exclude individual values from the statistics calculations. If the array is empty, all values will be used.

> **Exclude From Stats (F)** (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Pressure Results (Array) This array control contains pressure metrics results.

> **Pressure Results** (Cluster, Pressure Results.ctl) This control contains the basic pressure metrics calculations results.

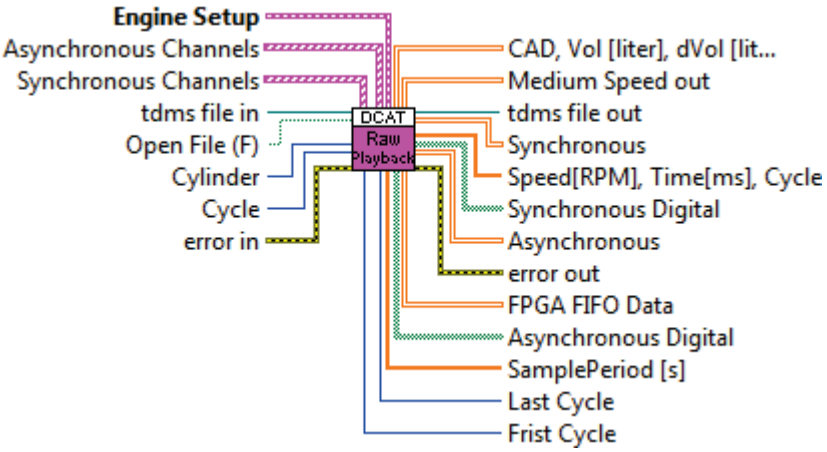
Pressure Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

> **Stats** (Cluster, Stats.ctl) This control contains calculated statistical results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Read Raw Cycle Data (normal priority) This VI uses the Synchronous and Asynchronous raw data to create a set of cycle arrays that are scaled and ready for processing. This VI should run one time with the TDMS file to open and initialize the file and then it can be run with the cylinder and cycle inputs to return a single cylinder and cycle.



Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous Digital (Array, Asynchronous Digital.ctl) Asynchronous Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

Digital Sample (Boolean) Digital Sample

Asynchronous Measurements (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

CAD, Vol [liter], dVol [liter], Area [m2] (Array, VolumeArrays.ctl) This indicator contains the data generated by the volume calculations or the volume file.

(Single Float) CAD, Volume, dVolume, Area

Cycle (U32, Cycle.ctl) Cylinder associated with the current data set. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

FPGA FIFO Data (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

- **FPGA Data** (Double Float) FPGA Data

First Cycle (U32) This control displays the cycle number of the first available cycle.

Last Cycle (U32) This control displays the cycle number of the last available cycle.

Medium Speed out (Array, Medium Speed sgl.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Medium Speed Sample

Open File (F) (Boolean) When true, the TDMS file will be opened and initialized.

SamplePeriod [s] (Array, SamplePeriod.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

- **Waveform** (Single Float) Sample Period

Speed [RPM], Time [ms], Cycle (Array, STC.ctl) This array control contains the speed, time, and cycle. The 0 index in the array is always speed. The 1 index is always time. The 2 index is always cycle number. This array is only used by uDCAT.

- **Value** (Double Float) Speed, Time, or Cycle

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Synchronous Sample

Synchronous Digital (Array, Synchronous Digital.ctl) Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

- **Digital Sample** (Boolean) Digital Sample

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single

measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

TDMS file in (Refnum) This control is a standard TDMS file reference.

TDMS file out (Refnum) This control is a standard TDMS file reference.

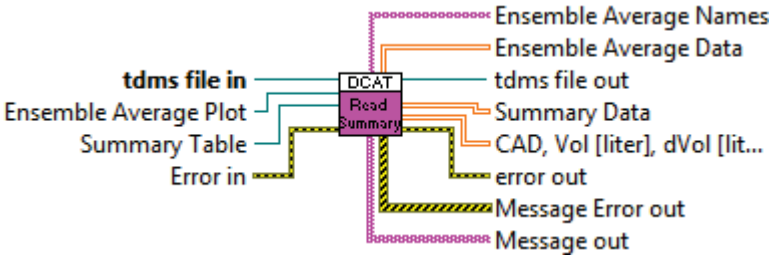
error in (Cluster) This control is a standard LabVIEW error cluster.

error out (Cluster) This control is a standard LabVIEW error cluster.

Read Summary from TDMS (normal priority) This VI reads the summary data files. The data is presented as a 2 or 3 dimensional array.

2D Arrays:
Index 0 = Cylinder
Index 1 = Cycle

3D Arrays:
Index 0 = Cylinder
Index 1 = Parameter
Index 2 = Cycle



CAD, Vol [liter], dVol [liter], Area [m2] (Array, VolumeArrays.ctl) This indicator contains the data generated by the volume calculations or the volume file.

(Single Float) CAD, Volume, dVolume, Area

Ensemble Average Data (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Ensemble Average Names (Array) This control contains the synchronous data names.

Synchronous Data Name (String) Synchronous Data Name

Ensemble Average Plot (Refnum) This control is a refnum of a waveform plot of ensemble average data.

Error in (Cluster) This control is a standard LabVIEW error cluster.

Message Error out (Array) This control contains error messages recorded with the data.

Message Error out (Cluster) This control is the standard LabVIEW error cluster.

Messages (Array) Array of messages recorded in the summary file.

Message (String) Calculation errors and messages.

Summary Data (Array) This control contains the summary data in the file.

Summary Data (Single Float) Summary Data

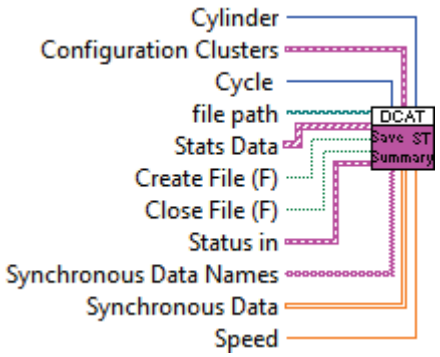
Summary Table (Refnum) This control is a refnum of a table to be filled with a summary data.

TDMS file in (Refnum) This control is a standard TDMS file reference.

TDMS file out (Refnum) This control is a standard TDMS file reference.

error out (Cluster) This control is a standard LabVIEW error cluster.

Save Summary From Stream (normal priority) This VI saves the results of the stats calculations to a TDMS file. This VI must run once to create a file and once to close the file. The "Configuration Cluster" and "File Path" are only required to be wired when the file is created. Summary files are created in the same way that the Real Time summary files are created.



Close File (F) (Boolean) This control should be set true when the file is to be closed. It should remain false while creating the file and saving data to the file.

Configuration Clusters (Cluster, Configuration Clusters.ctl) This control contains a cluster of all the setup clusters. It is used to pass all of the setup clusters to the saving VIs.

Create File (F) (Boolean) This control should be set true when a new file is to be created. It should remain false when saving data and closing the file.

Cycle (U32, Cycle.ctl) Cylinder associated with the current data set. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

File Path (Path) This control defines the file name and path.

Stats Data (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

> **Stats** (Cluster, Stats.ctl) This control contains calculated statistical results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous Data (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

> **Synchronous Sample** (Single Float) Synchronous Sample

Synchronous Data Names (Array) This control contains the synchronous data names.

> **Synchronous Data Name** (String) Synchronous Data Name

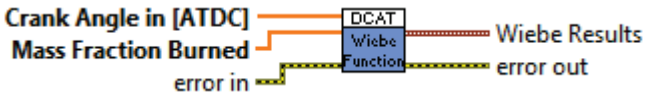
Wiebe Parameters (normal priority) This VI calculates the Wiebe Parameters for a cumulative heat release profile. The cumulative heat release profile is defined by the following equation.

x = 1 - exp(- a ((Theta - Theta_soc) / dTheta) ^ (m + 1))

x = mass fraction burned
Theta = Crank Angle
Theta_soc = Crank Angle @ Start of Combustion

dTheta = Duration of Combustion
a and m = Wiebe Coefficients

The input arrays to this VI should begin at the start of combustion and finish at the end of combustion. If no solution can be found, the VI will return values of a = 5 and m = 2. This VI should not run in the real time loop because it contains an iterative solver that may take some time to converge.



Crank Angle [ATDC] (Array, Crank Angle.ctl) Crank angle values to complement the synchronous arrays. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Crank Angle [CAD] (Single Float) Crank Angle associated with a specific element in the sampled array.

Mass Fraction Burned (Array, Mass Fraction Burned.ctl) Mass fraction burned as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Mass Fraction Burned (Single Float) Mass Fraction Burned

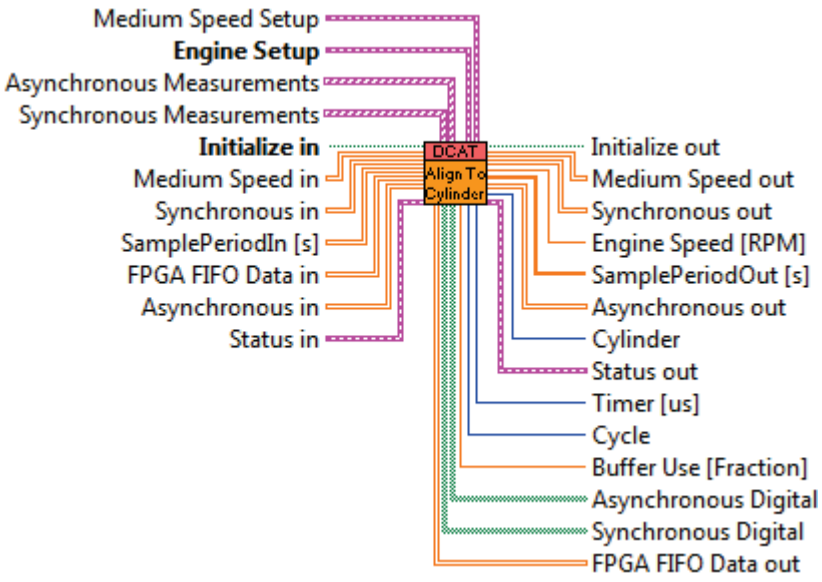
Wiebe Results (Cluster, Wiebe Results.ctl) This control contains the Wiebe function coefficients.

error in (Cluster) This control is the standard LabVIEW error cluster.

error out (Cluster) This control is the standard LabVIEW error cluster.

PreProcess

Align to Cylinder (normal priority) This VI takes the raw data streams and divides it up into individual cylinder events. This VI is to be called repeatedly and faster than each cylinder event. It will return a single cylinder's data when the complete cylinder arrays are available. If no new cylinder information is available, a cylinder number of -1 is returned. The VI will return an error if all the information cannot be processed but the VI will attempt to recover. This VI also ensemble averages the waveforms and applies the top dead center fine adjustment from the engine setup cluster. This VI also ensemble averages the data if enabled and windows the data if enabled. The Medium Speed Data filtering is also applied in this VI.



Asynchronous Digital (Array, Asynchronous Digital.ctl) Asynchronous Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

Digital Sample (Boolean) Digital Sample

Asynchronous Measurements (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Asynchronous in (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous out (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Buffer Use [Fraction] (Single Float, Buffer Use.ctl) This control displays the fraction of the buffer or queue that is currently used.

Cycle (U32, Cycle.ctl) Cylinder associated with the current data set. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

FPGA FIFO Data in (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

FPGA Data (Double Float) FPGA Data

FPGA FIFO Data out (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

FPGA Data (Double Float) FPGA Data

Initialize in (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Initialize out (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed Setup (Cluster, Medium Speed Setup.ctl) This control contains the medium speed DAQ setup.

Medium Speed in (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

Medium Speed out (Array, Medium Speed sgl.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Medium Speed Sample

SamplePeriodIn [s] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

SamplePeriodOut [s] (Array, SamplePeriod.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

Waveform (Single Float) Sample Period

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous Digital (Array, Synchronous Digital.ctl) Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

Digital Sample (Boolean) Digital Sample

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Synchronous in (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

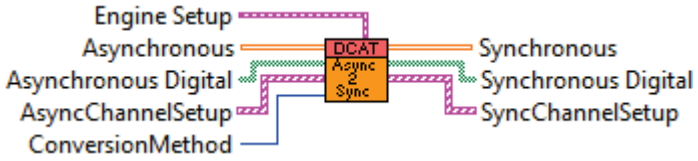
Synchronous Sample (Single Float) Synchronous Sample

Synchronous out (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Timer [us] (U32, Timer.ctl) Timer contains the current value of the FPGA free timer. The time is updated once every cycle.

Async2Sync (reentrant, subroutine) This VI converts all the Asynchronous data to Synchronous Data.



Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous Channels (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Asynchronous Digital (Array, Asynchronous Digital.ctl) Asynchronous Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

Digital Sample (Boolean) Digital Sample

ConversionMethod (Enum U16, Conversion Method.ctl) This control determines the method used to convert the asynchronous data to synchronous data.

0: Fast: This method uses a percentage of cycle to convert data.

1: Accurate: This method uses the reduced resolution to convert data.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Synchronous Digital (Array, Synchronous Digital.ctl) Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

Digital Sample (Boolean) Digital Sample

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Filter Top Level (normal priority) This VI filters the signals based on one of several methods. This VI filters up to 16 Cylinders of Data and 16 different filter definitions.

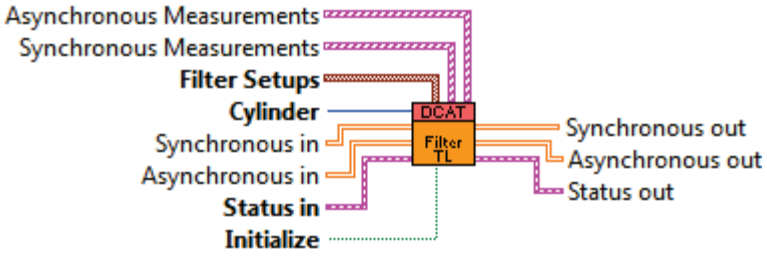
None: Performs no filtering to the signal.

Rolling N Cycles: Averages last N Cycles ever cycle (Slow).

Forward and Reverse IIR: Filtering with a 0 phase shift but shrinks the data set.

FIR Filter w/ Rollback: Filters data and then applies a negative phase shift.

Boxcar Average: Averages N points before and after every point for the value of the point.



Asynchronous Measurements (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Asynchronous in (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the

array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous out (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Filter Setups (Array, Filter Setups.ctl) This control contains the array of filter setup clusters. Each element in the array contains the setup information for a different filter.

Filter Setup (Cluster, Filter Setup.ctl) This control contains the filter setup.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

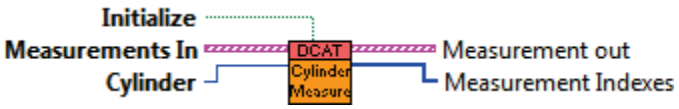
Synchronous in (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Synchronous out (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Get Cylinder Measurements (reentrant, normal priority) This VI separates and stores the channel configuration for each cylinder.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Measurement Indexes (Array) This control contains the indexes of the measurements associated with current cylinder. The indexes are based on the "Measureemtns In" and not the "Measurements out".

Measurement Index (I32) Measurement Index

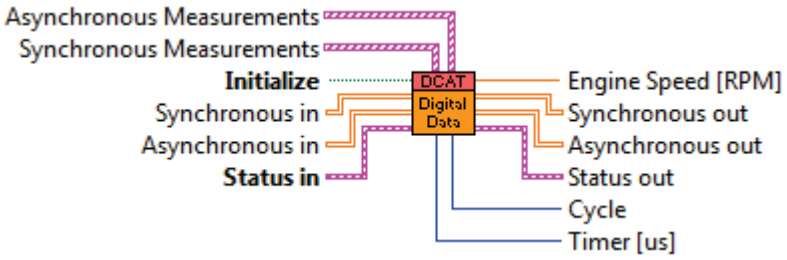
Measurement out (Array) This control is an array of measurement setup clusters defining all the measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Channel Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Measurements In (Array) This control is an array of measurement setup clusters defining all the measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Channel Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

GetStreamedDigitalData (reentrant, normal priority) This VI extracts the digital data (Speed, Time, and Cycle) from the streaming data.



Asynchronous Measurements (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Asynchronous in (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample. This control is defined by "Asynchronous.ctl".

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous out (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample. This control is defined by "Asynchronous.ctl".

Asynchronous Sample (Single Float) Asynchronous Sample

Cycle (U32, Cycle.ctl) Cylinder associated with the current data set. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single

measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Synchronous in (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample. This control is defined by "Synchronous.ctl".

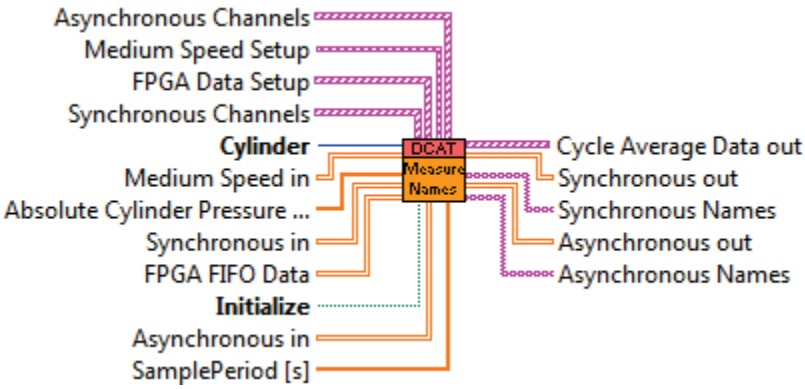
Synchronous Sample (Single Float) Synchronous Sample

Synchronous out (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample. This control is defined by "Synchronous.ctl".

Synchronous Sample (Single Float) Synchronous Sample

Timer [us] (U32, Timer.ctl) Timer contains the current value of the FPGA free timer. The time is updated once every cycle.

Measurement Names (normal priority) This VI generates a list of unique names for each channel based on the name given to the measurement. The unique names are required to log them to the summary file. A number is appended to measurements with duplicate names. The Units are also appended to the name. This VI also replaces the cylinder pressure with the pegged cylinder pressure if available.



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Asynchronous Channels (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Asynchronous Names (Array) This array control contains a list of unique names for the asynchronous measurements.

Asynchronous Name (String) Asynchronous Name

Asynchronous in (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous out (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Cycle Average Data out (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

FPGA Data Setup (Array, FPGA Data Setup.ctl) This control contains the FPGA data stream setup.

FPGA FIFO Data Configuration (Cluster, FPGA Data Configuration.ctl) This control contains the FPGA Data configuration setup.

FPGA FIFO Data (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

FPGA Data (Double Float) FPGA Data

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed Setup (Cluster, Medium Speed Setup.ctl) This control contains the medium speed DAQ setup.

Medium Speed in (Array, Medium Speed sgl.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Medium Speed Sample

SamplePeriod [s] (Array, SamplePeriod.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

- **Waveform** (Single Float) Sample Period

Synchronous Channels (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

- **Measurement Setup** (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Synchronous Names (Array) This array control contains a list of unique names for the synchronous measurements.

- **Synchronous Name** (String) Synchronous Name

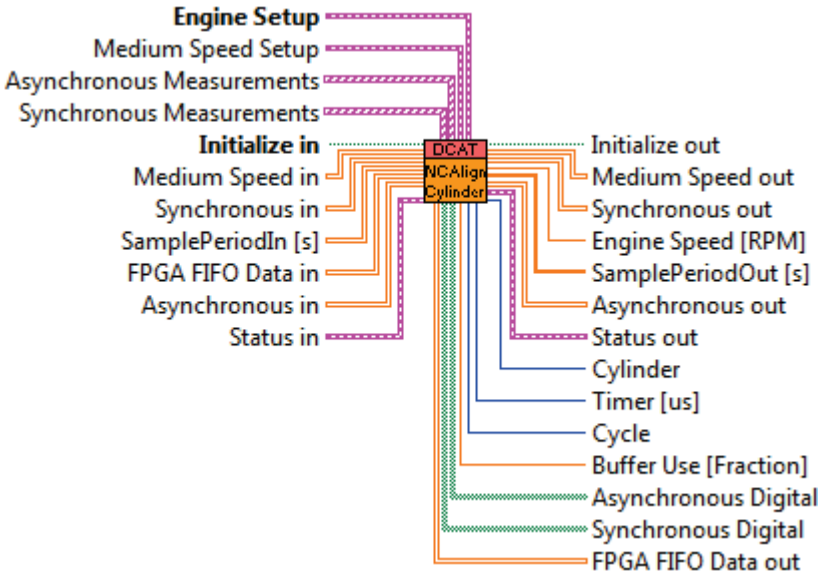
Synchronous in (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Synchronous Sample

Synchronous out (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Synchronous Sample

Next Cycle Align to Cylinder (normal priority) This VI takes the raw data streams and divides it up into individual cylinder events. This VI is to be called repeatedly and faster than each cylinder event. It will return a single cylinder's data when the complete cylinder arrays are available. If no new cylinder information is available, a cylinder number of -1 is returned. The VI will return an error if all the information cannot be processed but the VI will attempt to recover. This VI also ensemble averages the data if enabled and windows the data if enabled. The Medium Speed Data filtering is also applied in this VI.



Asynchronous Digital (Array, Asynchronous Digital.ctl) Asynchronous Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

Digital Sample (Boolean) Digital Sample

Asynchronous Measurements (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Asynchronous in (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous out (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the

array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Buffer Use [Fraction] (Single Float, Buffer Use.ctl) This control displays the fraction of the buffer or queue that is currently used.

Cycle (U32, Cycle.ctl) Cylinder associated with the current data set. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

FPGA FIFO Data in (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

FPGA Data (Double Float) FPGA Data

FPGA FIFO Data out (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

FPGA Data (Double Float) FPGA Data

Initialize in (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Initialize out (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed Setup (Cluster, Medium Speed Setup.ctl) This control contains the medium speed DAQ setup.

Medium Speed in (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

Medium Speed out (Array, Medium Speed sgl.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Medium Speed Sample

SamplePeriodIn [s] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

- **SamplePerid [s] / Time [ms]** (Double Float) Sample Period [s] / Time [ms]

SamplePeriodOut [s] (Array, SamplePeriod.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

- **Waveform** (Single Float) Sample Period

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous Digital (Array, Synchronous Digital.ctl) Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

- **Digital Sample** (Boolean) Digital Sample

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

- **Measurement Setup** (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Synchronous in (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Synchronous Sample

Synchronous out (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Timer [us] (U32, Timer.ctl) Timer contains the current value of the FPGA free timer. The time is updated once every cycle.

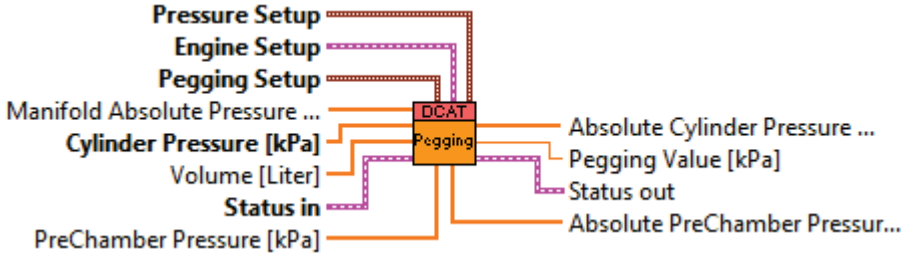
Pegging (normal priority) This VI pegs a single channel according to one of several methods. If an average length values is provided, the average is centered on the location.

None: Does not process the array.

Constant: Uses a constant location, average length, and pegging value to peg the cycle.

Synchronous MAP: Uses a MAP Channel at the specified location and averaging length.

Polytropic: Compares the Pressure and Volume over the PolyC calculation window to determine the cylinder Pegging Value that will match the specified PolyC Value.



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Absolute PreChamber Pressure [kPa abs] (Array, Absolute PreChamber Pressure.ctl) Absolute prechamber pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute PreChamber Pressure [kPa abs] (Single Float) Absolute PreChamber Pressure

Cylinder Pressure [kPa] (Array, Cylinder Pressure.ctl) Cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Cylinder Pressure [kPa] (Single Float) Cylinder Pressure

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Manifold Absolute Pressure [kPa abs] (Array, Manifold Absolute Pressure.ctl) Manifold Absolute Pressure as a function of crank angle.

Manifold Absolute Pressure [kPa abs] (Single Float) Manifold Absolute Pressure

Pegging Setup (Cluster, Pegging Setup.ctl) This control contains the Pegging setup.

Pegging Value [kPa] (Single Float) This control reports the value added to the cylinder pressure to convert it to an absolute pressure.

PreChamber Pressure [kPa] (Array, PreChamber Pressure.ctl) PreChamber pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

PreChamber Pressure [kPa] (Single Float) PreChamber Pressure

Pressure Setup (Cluster, Pressure Setup.ctl) This control contains the basic pressure metrics calculations setup.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Volume [Liter] (Array, Volume.ctl) Calculated volume of the cylinder as a function of the crank angle.

Volume [Liter] (Single Float) Cylinder Volume

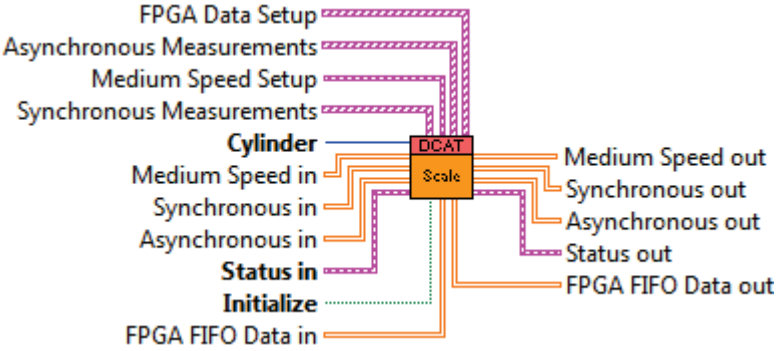
Scale (reentrant, normal priority) This VI scales the raw signals to engineering units based on one of several methods.

Gain Only: Multiplies the signals by the gain values. (Fast)

Gain and Offset: Multiplies the signals by the gain values and adds the offset values. (Fast)

Table: Uses the lookup tables to determine final value. (Slow)

Polynomial: Uses a function in the form of y=f(x) (ex. "2*x*exp(2*x)+1") to determine the final value. (Slower)



Asynchronous Measurements (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

> **Measurement Setup** (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Asynchronous in (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

> **Asynchronous Sample** (Single Float) Asynchronous Sample

Asynchronous out (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

> **Asynchronous Sample** (Single Float) Asynchronous Sample

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

FPGA Data Setup (Array, FPGA Data Setup.ctl) This control contains the FPGA data stream setup.

> **FPGA FIFO Data Configuration** (Cluster, FPGA Data Configuration.ctl) This control contains the FPGA Data configuration setup.

FPGA FIFO Data in (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

> **FPGA Data** (Double Float) FPGA Data

FPGA FIFO Data out (Array, FPGA FIFO Data.ctl) FPGA Data array contains the samples recorded by the FPGA. The first dimension in the array

corresponds to the Time [ms], ID, Data . The second dimension in the array corresponds to each sample.

FPGA Data (Double Float) FPGA Data

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed Setup (Cluster, Medium Speed Setup.ctl) This control contains the medium speed DAQ setup.

Medium Speed in (Array, Medium Speed sgl.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Medium Speed Sample

Medium Speed out (Array, Medium Speed sgl.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Medium Speed Sample

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Synchronous in (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

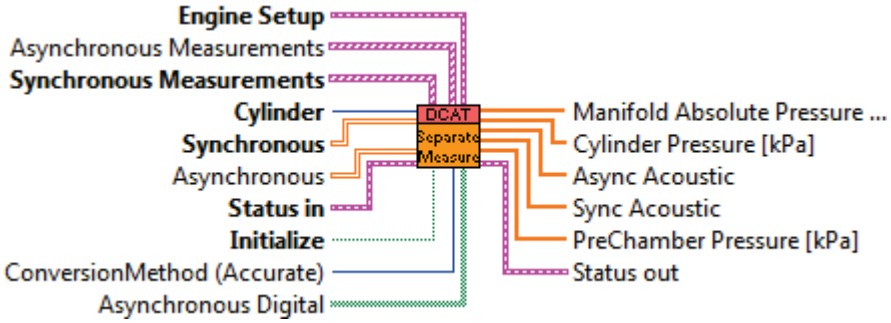
Synchronous Sample (Single Float) Synchronous Sample

Synchronous out (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the

array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Separate Measurements (reentrant, normal priority) This VI separates the manifold air pressure, cylinder pressure, and prechamber pressure from the synchronous array. An error occurs if no cylinder pressure is found. A warning occurs if multiple cylinder pressures are found. The first array is used if multiple signals of any kind are found.



Async Acoustic (Array, Asynchronous Acoustic.ctl) Asynchronous acoustic data as a function of crank angle. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Asynchronous Acoustic (Single Float) Asynchronous Acoustic Measurement

Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous Digital (Array, Asynchronous Digital.ctl) Asynchronous Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

Digital Sample (Boolean) Digital Sample

Asynchronous Measurements (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

ConversionMethod (Accurate) (Enum U16, Conversion Method.ctl) This control specifies the method used to convert asynchronous data to synchronous data.

0: Fast: Uses the percentage of the cycle to convert the data.

1: Accurate: Uses the reduced resolution encoder to convert the data.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Cylinder Pressure [kPa] (Array, Cylinder Pressure.ctl) Cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Cylinder Pressure [kPa] (Single Float) Cylinder Pressure

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Manifold Absolute Pressure [kPa abs] (Array, Manifold Absolute Pressure.ctl) Manifold absolute pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Manifold Absolute Pressure [kPa abs] (Single Float) Manifold Absolute Pressure

PreChamber Pressure [kPa] (Array, PreChamber Pressure.ctl) PreChamber pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

PreChamber Pressure [kPa] (Single Float) PreChamber Pressure

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Sync Acoustic (Array, Synchronous Acoustic.ctl) Synchronous acoustic data as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Synchronous Acoustic (Single Float) Synchronous Acoustic Measurement

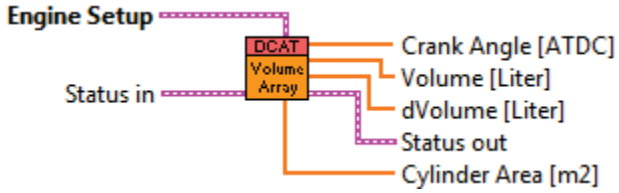
Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Volume Array (normal priority) This VI generates an array of CAD positions to match the encoder resolution and engine setup. Matching cylinder volume and volume difference array are also created. The volume difference is created by calculating the volume half way between the CAD measurements. This VI also calculates the surface area of the cylinder at each crank angle. Two sets of calculations are used, one if the pin offset is set to zero (Faster) and another is the pin offset is not zero (Slower). The arrays may also be created from a file containing the Crank Angle, Volume, dVolume and Cylinder Area in each column. The data should be separated by a comma. The data should cover an entire engine cycle. The data will be interpolated if the data is not at the correct resolution.



Crank Angle [ATDC] (Array, Crank Angle.ctl) Crank angle associated with each element in the sampling resolution.

Crank Angle [CAD] (Single Float) Crank Angle associated with a specific element in the sampled array.

Cylinder Area [m2] (Array, Cylinder Area.ctl) Calculated area of the cylinder walls including the piston and the head as a function of the crank angle.

Cylinder Area [m2] (Single Float) Cylinder Area

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Volume [Liter] (Array, Volume.ctl) Calculated cylinder volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

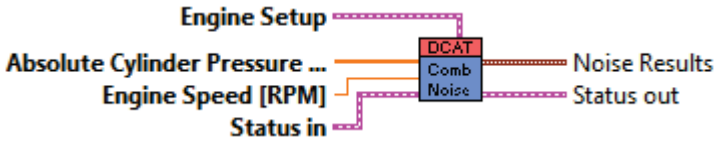
Volume [Liter] (Single Float) Cylinder Volume

dVolume [Liter] (Array, dVolume.ctl) Differential volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

dVolume [Liter] (Single Float) Differential Volume

Process

Combustion Noise (reentrant, normal priority) This VI calculates the combustion noise values. The VI uses cylinder pressure to estimate the combustion noise. The calculation method should produce values similar to the Lucas Noise Meter but has not been tested an engine in detail. (Measurement of Combustion Noise in Diesel Engines by A. J. Herbert, M. F. Russell)



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

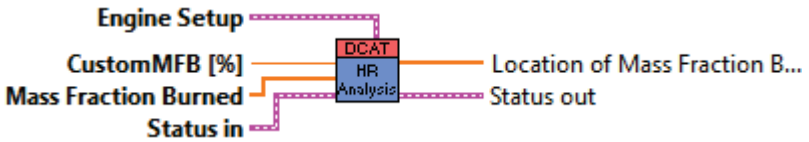
Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

Noise Results (Cluster, Combustion Noise Results.ctl) This control contains the results of the combustion noise calculations.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Custom HR Analysis (reentrant, subroutine) Calculates the location of a user specified Mass Fraction Burned from the Cumulative Heat Release.



Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Location of Mass Fraction Burned [CAD] (Array) This control displays the location of the input mass fraction burned.

Location of Mass Fraction Burned [CAD] (Single Float) Location of Mass Fraction Burned

Mass Fraction Burned (Single Float) This control specifies the desired mass fraction burned value.

Mass Fraction Burned (Array, Mass Fraction Burned.ctl) Mass fraction burned as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Mass Fraction Burned (Single Float) Mass Fraction Burned

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

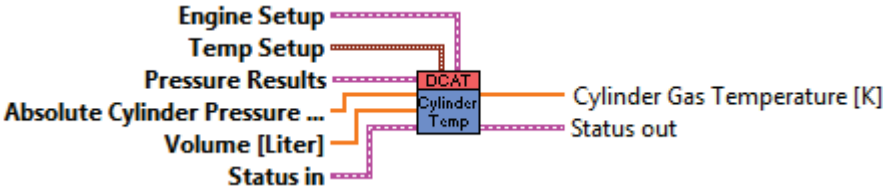
Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Cylinder Temp (reentrant, subroutine) This VI calculates the cylinder temperatures based on the Ideal Gas Law. Intake and exhaust temperature and

pressure must be known along with the valve timing. The exhaust conditions are only needed if the Residual Fraction is used.

T = (Tivc / (Pivc * Vivc)) * P * V
Tivc = Ti * (1 - f) + 1.39 * Te * (Pi - Pe)^(n - 1 / n) * f

T = Instantaneous Cylinder Temperature [K]
Tivc = Cylinder Temperature at Intake Valve Closing [K]
P = Instantaneous Cylinder Pressure [kPa]
Pivc = Cylinder Pressure at Intake Valve Closing [kPa]
V = Instantaneous Cylinder Volume [liter]
Vivc = Cylinder Volume at Intake Valve Closing [liter]
Ti = Intake Manifold Temperature [K]
Te = Exhaust Manifold Temperature [K]
Pi = Intake Manifold Pressure [kPa]
Pe = Exhaust Manifold Pressure [kPa]
f = Residual Fraction []



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Cylinder Gas Temperature [K] (Array, Cylinder Gas Temperature.ctl) Cylinder Gas Temperature Estimate.

Cylinder Gas Temperature [K] (Single Float) Cylinder Gas Temperature

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Pressure Results (Cluster, Pressure Results.ctl) This control contains the basic pressure metrics calculations results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

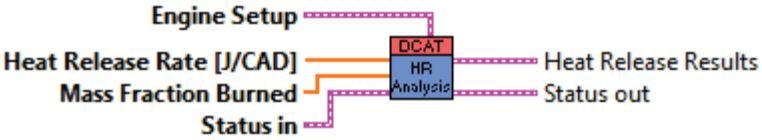
Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Temp Setup (Cluster, Temp Setup.ctl) This control contains the cylinder temperature calculation setup.

Volume [Liter] (Array, Volume.ctl) Calculated cylinder volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Volume [Liter] (Single Float) Cylinder Volume

HR Analysis (reentrant, subroutine) Calculates the location of 5%, 10%, 50%, 75%, and 90% Mass Fraction Burned. This VI also calculates the Location of Maximum Heat Release Rate and the Maximum Heat Release Rate.



Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Heat Release Rate [J/CAD] (Array, Heat Release Rate.ctl) Heat release rate as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. Covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Heat Release Rate [J/CAD] (Single Float) Heat Release Rate

Heat Release Results (Cluster, Heat Release Results.ctl) This control contains the heat release calculations results.

Mass Fraction Burned (Array, Mass Fraction Burned.ctl) Mass fraction burned as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Mass Fraction Burned (Single Float) Mass Fraction Burned

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

HR Raw (reentrant, subroutine) This VI calculates the heat release rate and the cumulative heat release based on one of several methods. (Heywood, Internal Combustion Fundamentals, McGraw-Hill, 1988; Grimm, Johnson, Review of Simple Heat Release Computations, SAE Technical Paper Series 900445, 1990)

Single Zone: Uses the single zone model for heat transfer to determine the heat release.

dQ/dTheta = (1 / (n - 1)) * V * dP/dTheta + (n / (n - 1)) * P * dV/dTheta

Single Zone Dual Transducer: Uses the single zone model for heat transfer to determine the heat release.

dQ/dTheta = (1 / (n - 1)) * (Vmc * dPmc/dTheta + Vpc * dPpc/dTheta) + (n / (n - 1)) * P * dV/dTheta

Single Zone + Heat Transfer: Uses the single zone model for heat transfer to determine the heat release.

dQ/dTheta = (1 / (n - 1)) * V * dP/dTheta + (n / (n - 1)) * P * dV/dTheta + dQht/dTheta

dQht/dTheta = alpha * h * A_Wall * (T - T_Wall)

Single Zone Dual Transducer + Heat Transfer: Uses the single zone model for heat transfer to determine the heat release.

dQ/dTheta = (1 / (n - 1)) * (Vmc * dPmc/dTheta + Vpc * dPpc/dTheta) + (n / (n - 1)) * P * dV/dTheta + dQht/dTheta

dQht/dTheta = alpha * h * A_Wall * (T - T_Wall)

Modified Rassweiler and Withrow:

dQ/dTheta = (1 / (n - 1)) * V * dPc/dTheta

dPc/dTheta = Pj - Pi * (Vi / Vj)^n

Pressure Ratio:

PR = P / Pmot

Pmot = P1 * (V1 / V)^n

n = Specific Heat Ratio []
dQ/dTheta = Heat Release Rate [J / CAD]
V = Instantaneous Volume [liter]
Vmc = Instantaneous Main Chamber Volume [liter]
Vpc = Pre-Chamber Volume [liter]
P = Instantaneous Pressure [kPa]
P1 = Reference Pressure [kPa]
Pmc = Instantaneous Main Chamber Pressure [kPa]
Pmot = Motored Pressure [kPa]
Ppc = Instantaneous Pre-Chamber Pressure [kPa]
PR = Pressure Ratio []
dP = Differential Pressure [kPa]
dTheta = Differential Crank Angle [CAD]

dV = Differential Volume [liter]
dPmc = Differential Main Chamber Pressure [kPa]
dPpc = Differential Pre-Chamber Pressure [kPa]
dVmc = Differential Main Chamber Volume [kPa]
j = Current Crank Angle [CAD]
i = Previous Crank Angle [CAD]
V1 = Reference Volume [liter]
Vi = Volume at Previous Crank Angle [liter]
Vj = Volume at Current Crank Angle [liter]
Pi = Pressure at Previous Crank Angle [kPa]
Pj = Pressure at Current Crank Angle [kPa]
alpha = Scaling Factor [Fraction]
h = Heat Transfer Coefficient [J/CAD-m2-K]
A_Wall = Wall Area at Each Crank Angle [m2]
T = Cylinder Gas Temperature [K]
T_Wall = Cylinder Wall Temperature [K]

The heat release calculations using heat transfer require a heat transfer coefficient. The Heat Transfer Coefficient is calculated by one of Several Methods. (Heywood, Internal Combustion Fundamentals, McGraw-Hill, 1988;)

Constant: Constant User Specified Value.

Woschni 1987: Woschni 1987 Correlation.

Woschni 1990: Modified Woschni Correlation to Better Model Partial Loads Conditions.

Hohenberg: Hohenberg Correlation.

The heat release calculations using heat transfer also require a cylinder wall temperature. This VI generates a Wall Temperature value for each CAD based on the Following.

Constant: Creates an Array of Temperature equal to T_wall [K]

Linear: Creates an Array of Temperatures equal to
T_wall [K] = T_TDC [K] * (1 - exp(- c * x)) / (x * c)
c = ln(T_TDC [K] / T_BDC [K])

T_Wall = Wall Temperature [K]
T_TDC = Wall Temperature at TDC [K]
T_BDC = Wall Temperature at BDC [K]
x = Relative Stroke [Fraction]
c = Intermediate Value

The Start of Combustion is Base on One of Several Methods.

Constant: User Defined Start of Combustion.

N Below Zero: Starts the Heat Release Cumulative Fraction when the Heat Release Contains N Consecutive Values Below Zero after the Starting Crank Angle.

Boxcar Average Below Zero: Start the Heat Release Cumulative Fraction when the +-N Point Boxcar Average of the Heat Release Value is Below Zero after the Starting Crank Angle.

N Above Zero: Starts the Heat Release Cumulative Fraction at the last Point where the Heat Release Contains N Consecutive Values Below Zero before TDC.

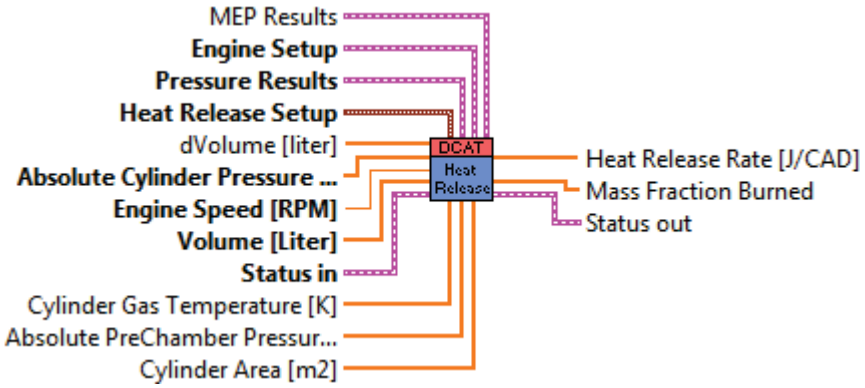
Boxcar Average Above Zero: Start the Heat Release Cumulative Fraction at the last Point where the +-N Point Boxcar Average of the Heat Release Value is Below Zero Before TDC.

The End of Combustion is Base on One of Several Methods.

Constant: User Defined End of Combustion.

N Below Zero: Ends the Heat Release Cumulative Fraction when the Heat Release Contains N Consecutive Values Below Zero Before the Ending Crank Angle.

Boxcar Average Below Zero: Ends the Heat Release Cumulative Fraction when the +-N Point Boxcar Average of the Heat Release Value is Below Zero Before the Ending Crank Angle.



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Absolute PreChamber Pressure [kPa abs] (Array, Absolute PreChamber Pressure.ctl) Absolute prechamber pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

- **Absolute PreChamber Pressure [kPa abs]** (Single Float) Absolute PreChamber Pressure

Cylinder Area [m2] (Array, Cylinder Area.ctl) Calculated area of the cylinder walls including the piston and the head as a function of the crank angle.

- **Cylinder Area [m2]** (Single Float) Cylinder Area

Cylinder Gas Temperature [K] (Array, Cylinder Gas Temperature.ctl) Cylinder Gas Temperature Estimate.

- **Cylinder Gas Temperature [K]** (Single Float) Cylinder Gas Temperature

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

Heat Release Rate [J/CAD] (Array, Heat Release Rate.ctl) Heat release rate as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. Covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

- **Heat Release Rate [J/CAD]** (Single Float) Heat Release Rate

Heat Release Setup (Cluster, Heat Release Setup.ctl) This control contains the heat release calculations configuration.

MEP Results (Cluster, MEP Results.ctl) This control contains the MEP calculations results.

Mass Fraction Burned (Array, Mass Fraction Burned.ctl) Mass fraction burned as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

- **Mass Fraction Burned** (Single Float) Mass Fraction Burned

Pressure Results (Cluster, Pressure Results.ctl) This control contains the basic pressure metrics calculations results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Volume [Liter] (Array, Volume.ctl) Calculated cylinder volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Volume [Liter] (Single Float) Cylinder Volume

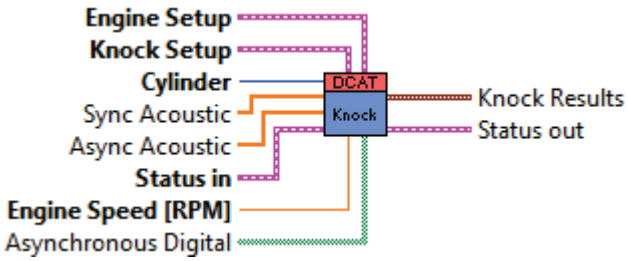
dVolume [liter] (Array, dVolume.ctl) Differential volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

dVolume [Liter] (Single Float) Differential Volume

Knock (reentrant, normal priority) This VI calculates knock based on the one of several methods. The knock calculations can use a synchronous or asynchronous signals.

BRIC: Bandpass, Rectify, Integrate, Compare

The BRIC method uses a bandpass filter to isolate a range of knock frequencies and then rectifies and integrates the signal to determine the knock intensity value. The knock intensity values are calculated for a reference and a knock window. The two windows are compared to determine the final knock intensity value.



Async Acoustic (Array, Asynchronous Acoustic.ctl) This control contains the asynchronous waveform. It is only used when the source of the calculations is the asynchronous measurements.

Asynchronous Acoustic (Single Float) Asynchronous Acoustic Measurement

Asynchronous Digital (Array, Asynchronous Digital.ctl) Asynchronous Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

Digital Sample (Boolean) Digital Sample

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

Knock Results (Cluster, Knock Results.ctl) This control contains the knock calculations results.

Knock Setup (Cluster, Knock Setup.ctl) This control contains the knock calculation setup.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

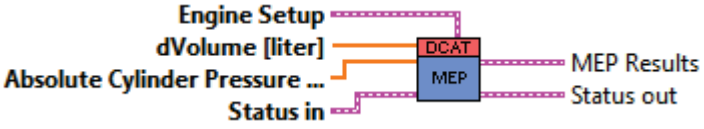
Sync Acoustic (Array, Synchronous Acoustic.ctl) This control contains the synchronous waveform. It is only used when the source of the calculations is the synchronous measurements.

Synchronous Acoustic (Single Float) Synchronous Acoustic Measurement

MEP (normal priority) This VI calculates the mean effective pressure values (Heywood, Internal Combustion Fundamentals, McGraw-Hill, 1988). NMEP is calculated for both a 2 and 4 stroke engine. PMEP and IMEP are only calculated for a 4 stroke engine.

IMEP = - Integral(P , dV) / Vd, From -180 to +180 [CAD]
PMEP = - Integral(P , dV) / Vd, From -360 to -180 and 180 to 360 [CAD]
NMEP = IMEP + PMEP

IMEP = Gross Indicated Mean Effective Pressure [kPa]
NMEP = Net Indicated Mean Effective Pressure [kPa]
PMEP = Pumping Indicated Mean Effective Pressure [kPa]
P = Instantaneous Pressure [kPa]
dV = Differential Volume [liter]
Vd = Displacement Volume [liter]



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

MEP Results (Cluster, MEP Results.ctl) This control contains the MEP calculations results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

dVolume [liter] (Array, dVolume.ctl) Differential volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

dVolume [Liter] (Single Float) Differential Volume

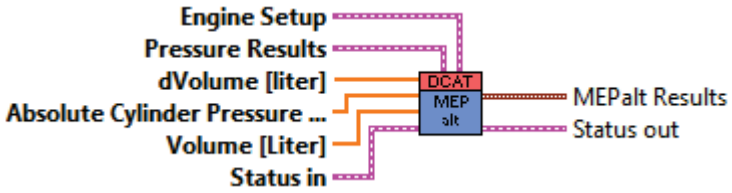
MEPalt (normal priority) This VI also calculates another variation of IMEP for a 4-stroke engine (SAE Technical Paper #04ANNUAL-411). It calculates the theoretical pressure of the cylinder after combustion and before ignition to determine the losses due to the valves. The extra valve losses are moved from the IMEP to the PMEP value.

IMEP = - Integral(P , dV) / Vd, From -180 to +180 [CAD]
PMEP = - Integral(P , dV) / Vd, From -360 to -180 and 180 to 360 [CAD]

PMEP1 = - Integral((P - Pi) , dV) / Vd, From 90 to 180 [CAD]
PMEP2 = - Integral((P - Pi) , dV) / Vd, From -180 to -90 [CAD]
Pi = P1 (V / V1)^n
PMEPalt = PMEP + PMEP1 + PMEP2
IMEPalt = IMEP + PMEP1 + PMEP2

IMEP = Gross Indicated Mean Effective Pressure [kPa]
PMEP = Pumping Indicated Mean Effective Pressure [kPa]
P = Instantaneous Pressure [kPa]
dV = Differential Volume [liter]

Vd = Displacement Volume [liter]
Pi = Instantaneous Ideal Pressure [kPa]
P1 = Reference Pressure [kPa]
V1 = Reference Volume [kPa]
n = Specific Heat Ratio []



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

MEPalt Results (Cluster, MEPalt Results.ctl) This control contains the MEP calculations results.

Pressure Results (Cluster, Pressure Results.ctl) This control contains the basic pressure metrics calculations results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Volume [Liter] (Array, Volume.ctl) Calculated cylinder volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Volume [Liter] (Single Float) Cylinder Volume

dVolume [liter] (Array, dVolume.ctl) Differential volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

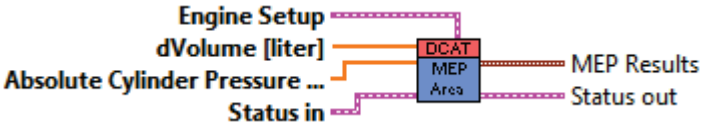
dVolume [Liter] (Single Float) Differential Volume

MEParea (normal priority) This VI also calculates another variations of IMEP for a 4-stroke engine (SAE Technical Paper #04ANNUAL-411). It calculates the IMEP and PMEP values for the areas of the curves that do not overlap. The overlapping sections of the diagram are ignored.

IMEP = - Integral(P , dV) / Vd, From -180 to +180 [CAD]
PMEP = - Integral(P , dV) / Vd, From -360 to -180 and 180 to 360 [CAD]

IMEParea = IMEP - Overlap
PMEParea = PMEP - Overlap

IMEP = Gross Indicated Mean Effective Pressure [kPa]
PMEP = Pumping Indicated Mean Effective Pressure [kPa]
P = Instantaneous Pressure [kPa]
dV = Differential Volume [liter]
Vd = Displacement Volume [liter]



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

MEP Results (Cluster, MEParea Results.ctl) This control contains the MEP calculations results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

dVolume [liter] (Array, dVolume.ctl) Differential volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

dVolume [Liter] (Single Float) Differential Volume

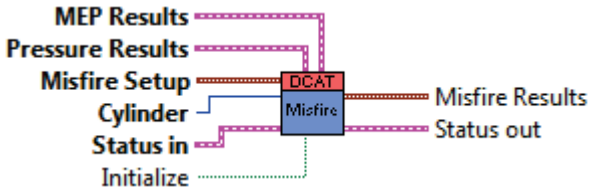
Misfire (reentrant, normal priority) This VI detects misfires based of the peak pressure or NMEP and a threshold pressure or a percentage of the average historic threshold. A misfire is reported if the peak pressure or NMEP drops below the threshold.

Peak Pressure (Absolute): Detects a misfire when the peak pressure drops below an absolute threshold.

Peak Pressure (% of Historic): Detects a misfire when the peak pressure drops below a percent of the historic average.

IMEP (Absolute): Detects a misfire when the NMEP drops below an absolute threshold.

IMEP (% of Historic): Detects a misfire when the NMEP drops below an absolute threshold.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

MEP Results (Cluster, MEP Results.ctl) This control contains the MEP calculations results.

Misfire Results (Cluster, Misfire Results.ctl) This control contains the results of the misfire calculations.

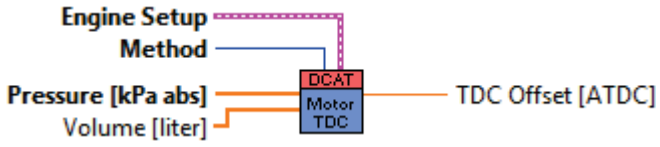
Misfire Setup (Cluster, Misfire Setup.ctl) This control contains the misfire setup.

Pressure Results (Cluster, Pressure Results.ctl) This control contains the basic pressure metrics calculations results.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Motoring TDC Check (reentrant, normal priority) This VI uses the cylinder pressure to determine the value of the TDC offset based on one of several methods.



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Method (Enum U32) This control determines the method used to calculate the additional top dead center offset needed to align the data to the proper location.

0: Peak Pressure: Simply reports the difference between the measured peak pressure and the TDC location.

1: Spline Interpolation: Same as the Peak Pressure method but uses a spline interpolation near TDC to estimate the peak pressure.

2: Half Peak Pressure: Finds the location of the pressure with 1/2 the value of the peak pressure on both sides of TDC and reports the difference between the center point and the TDC location.

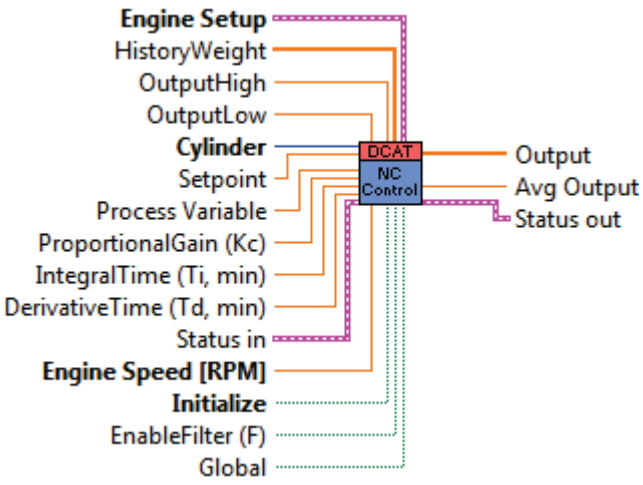
3: Most Probable: Uses several points on both sides of TDC to determine the most probable location of TDC.

TDC Offset [ATDC] (Single Float) This control reports the calculated TDC offset needed to align the cylinders.

Volume [liter] (Array, Volume.ctl) Calculated cylinder volume as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Volume [Liter] (Single Float) Cylinder Volume

NC Control (normal priority) This VI contains a set of PID controllers to calculate next cycle output values.



Avg Output (Single Float) This control contains the average value of PID outputs.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

DerivativeTime (Td, min) (Single Float) This control contains the derivative time for the PID controller.

EnableFilter (F) (Boolean) This control enables the process variable filter. The filter uses the history weight to bias the last few samples.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

Global (Boolean) This control sets the sub VI to run a single PID for all cylinders when true. When false, each cylinder uses its own PID.

HistoryWeight (Array) This control is used in the filtering of the inputs. The last several inputs are weighted with the current input.

- **History Weight** (Single Float) History Weight

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

IntegralTime (Ti, min) (Single Float) This control contains the integral time for the PID controller.

Output (Array) This control contains the value of the PID outputs for each cylinder.

- **Output** (Single Float) PID output

OutputHigh (Single Float) This control contains the output high is the maximum value for controller output.

OutputLow (Single Float) This control contains the output low is the minimum value for controller output.

Process Variable (Single Float) This control contains the process variable of the PID.

ProportionalGain (Kc) (Single Float) This control contains the proportional gain for the PID controller.

Setpoint (Single Float) This control contains the PID setpoint.

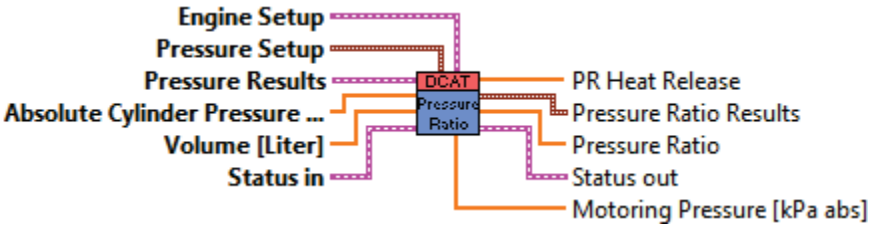
Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Pressure Ratio (reentrant, normal priority) This VI calculates the pressure ratio values (SAE Technical Paper Series #2000-01-0932). The pressure ratio is a simplified form of heat calculations similar to the Rassweiler and Withrow calculations. The calculations compare the measured cylinder pressure to a calculated motoring pressure. The ratio of these two signals is the pressure ratio.

PR = P / Pmot
Pmot = P1 * (V1 / V)^n
PRM10 = (PR(10) - 1) / (FPR - 1)
PRM25 = (PR(25) - 1) / (FPR - 1)
DILPAR = FPR + (0.5 * PRM10) - 1.275
MPR = FPR - 1

PR(10) = PR @ 10 CAD []
PR(25) = PR @ 25 CAD []
FPR = Final Pressure Ratio (Averaged over PolyE Calculation Window) []
PRM10 = Pressure Ratio Management Timing Parameter @ 25 CAD []
PRM25 = Pressure Ratio Management Timing Parameter @ 10 CAD []
DILPAR = Dilution Parameter []
MPR = Modified Pressure Ratio []
PR = Pressure Ratio []
P = Instantaneous Cylinder Pressure [kPa]
Pmot = Calculated Motoring Pressure [kPa]
P1 = Reference Pressure (at Start of PolyC Calculation) [kPa]
V = Instantaneous Cylinder Volume [liter]
V1 = Reference Volume (at Start of PolyC Calculation) [liter]
n = Specific Heat Ratio []



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Motoring Pressure [kPa abs] (Array, Motoring Pressure.ctl) Calculated motoring pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Motoring Pressure (Single Float) Motoring Pressure

PR Heat Release (Array, PR Heat Release Rate.ctl) Pressure ratio heat release rate as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

PR Heat Release Rate (Single Float) Pressure Ratio Heat Release Rate

Pressure Ratio (Array, Pressure Ratio.ctl) Pressure ratio as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Pressure Ratio (Single Float) Pressure Ratio

Pressure Ratio Results (Cluster, Pressure Ratio Results.ctl) This control contains the pressure ratio calculations results.

Pressure Results (Cluster, Pressure Results.ctl) This control contains the basic pressure metrics calculations results.

Pressure Setup (Cluster, Pressure Setup.ctl) This control contains the basic pressure metrics calculations setup.

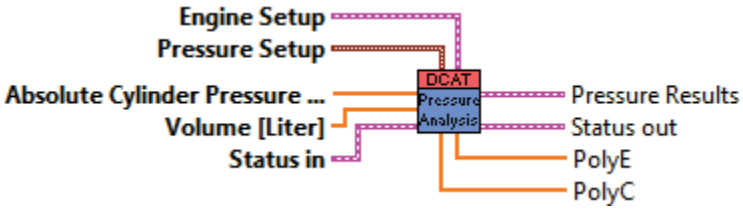
Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Volume [Liter] (Array, Volume.ctl) Calculated volume of the cylinder as a function of the crank angle.

Volume [Liter] (Single Float) Cylinder Volume

Pressure (reentrant, normal priority) This VI calculates the basic pressure metrics (cylinder peak pressure, the location of the cylinder peak pressure, the polytropic coefficient for expansion, the polytropic coefficient for compression, the exhaust pressure, and the intake pressure).



Absolute Cylinder Pressure [kPa abs] (Array, Absolute Cylinder Pressure.ctl) Absolute cylinder pressure as a function of crank angle. Each point in the array correlates to a sample in the sample resolution. This array covers 360 crank angle degrees for a 2-stroke engine and 720 crank angle degrees for a 4-stroke engine.

Absolute Cylinder Pressure [kPa abs] (Single Float) Absolute Cylinder Pressure

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

PolyC (Array, PolyC.ctl) Polytropic coefficient for compression as a function of crank angle for the polytropic coefficient for compression window. The number of samples is determined by the window size and the sampling resolution.

PolyC (Single Float) Polytropic Coefficient of Compression

PolyE (Array, PolyE.ctl) Polytropic coefficient for expansion as a function of crank angle for the polytropic coefficient for expansion window. The number of samples is determined by the window size and the sampling resolution.

PolyE (Single Float) Polytropic Coefficient of Expansion

Pressure Results (Cluster, Pressure Results.ctl) This control contains the basic pressure metrics calculations results.

Pressure Setup (Cluster, Pressure Setup.ctl) This control contains the basic pressure metrics calculations setup.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

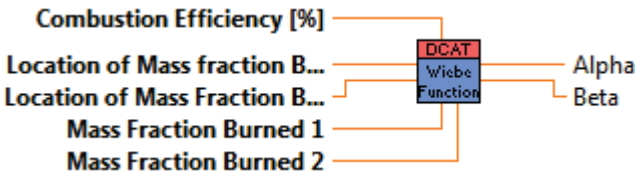
Volume [Liter] (Array, Volume.ctl) Calculated volume of the cylinder as a function of the crank angle.

Volume [Liter] (Single Float) Cylinder Volume

Wiebe Parameters RT (reentrant, subroutine) This VI calculates the Wiebe Parameters (alpha, beta) from a pair of burned duration (x), mass fraction burned (F) pairs. It also requires a combustion efficiency (eff).

alpha = ln(1 - eff)
beta = ln(ln(F2) / ln (F1)) / ln(x1 / x2)

Wiebe Function:
x = 1 - exp (- alpha * (F) ^ (beta + 1))



Alpha (Double Float) Wiebe Parameter Alpha

Beta (Double Float) Wiebe Parameter Beta

Combustion Efficiency [%] (Double Float) This control defines the combustion efficiency. The combustion efficiency is used to calculate alpha.

Location of Mass Fraction Burned 2 [CAD] (Double Float) This control defines the location of mass fraction burned of the second pair.

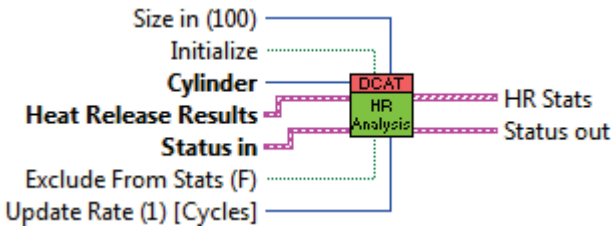
Location of Mass fraction Burned 1 [CAD] (Double Float) This control defines the location of mass fraction burned of the first pair.

Mass Fraction Burned 1 (Double Float) This control defines the mass fraction burned of the first pair.

Mass Fraction Burned 2 (Double Float) This control defines the mass fraction burned of the second pair.

Stats

HR Stats RT (reentrant, normal priority) This VI calculates the statistics for the calculated Heat Release values. This VI is good for up to 16 Cylinders.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

HR Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Heat Release Results (Cluster, Heat Release Results.ctl) This control contains the heat release calculations results.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

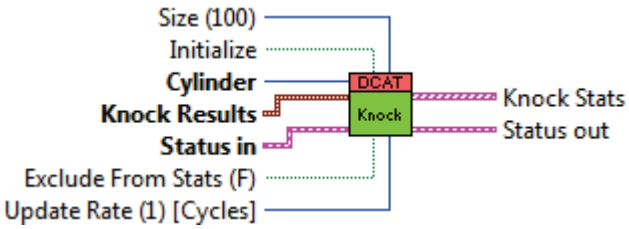
Size in (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Update Rate (1) [Cycles] (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

Knock Stats RT (reentrant, normal priority) This VI calculates the statistics for the calculated knock values. This VI is good for up to 16 Cylinders.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Knock Results (Cluster, Knock Results.ctl) This control contains the knock calculations results.

Knock Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

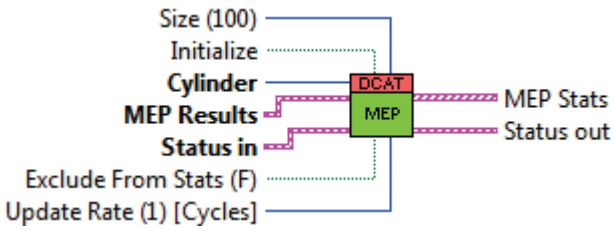
Size (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Update Rate (1) [Cycles] (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

MEP Stats RT (reentrant, normal priority) This VI calculates the statistics for the calculated mean effective pressure values. This VI is good for up to 16 Cylinders.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

MEP Results (Cluster, MEP Results.ctl) This control contains the MEP calculations results.

MEP Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

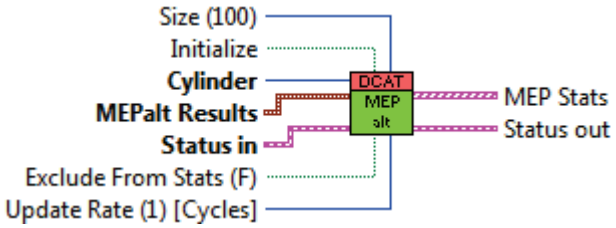
Size (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Update Rate (1) [Cycles] (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

MEPalt Stats RT (reentrant, normal priority) This VI calculates the statistics for the calculated mean effective pressure values. This VI is good for up to 16 Cylinders.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

MEP Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

MEPalt Results (Cluster, MEPalt Results.ctl) This control contains the MEP calculations results.

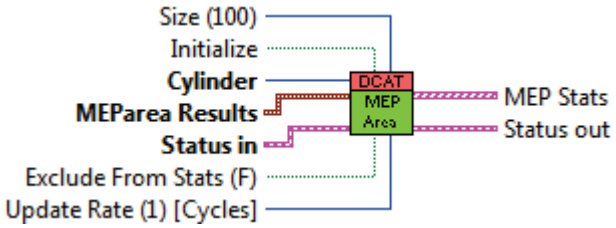
Size (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Update Rate (1) [Cycles] (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

MEParea Stats RT (reentrant, normal priority) This VI calculates the statistics for the calculated mean effective pressure values. This VI is good for up to 16 Cylinders.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

MEP Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

MEParea Results (Cluster, MEParea Results.ctl) This control contains the MEP calculations results.

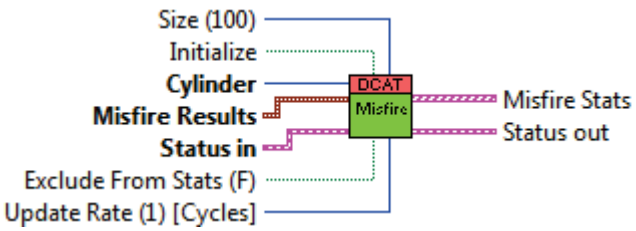
Size (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Update Rate (1) [Cycles] (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

Misfire Stats RT (reentrant, normal priority) This VI calculates the statistics for the calculated misfire values. This VI is good for up to 16 Cylinders.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Misfire Results (Cluster, Misfire Results.ctl) This control contains the results of the misfire calculations.

Misfire Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

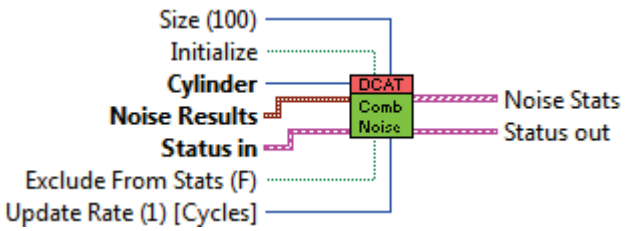
Size (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Update Rate (1) [Cycles] (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

Noise Stats RT (reentrant, normal priority) This VI calculates the statistics for the calculated combustion noise values. This VI is good for up to 16 Cylinders.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Noise Results (Cluster, Combustion Noise Results.ctl) This control contains the results of the combustion noise calculations.

Noise Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

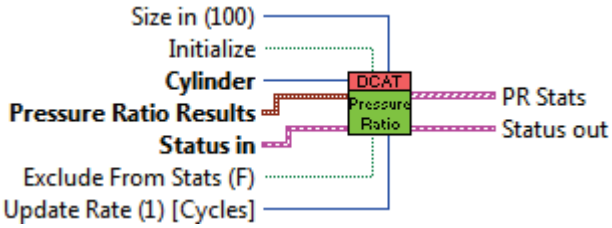
Size (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Update Rate (1) [Cycles] (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

Pressure Ratio Stats RT (reentrant, normal priority) This VI calculates the statistics for the calculated pressure ratio values. This VI is good for up to 16 Cylinders.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

PR Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Pressure Ratio Results (Cluster, Pressure Ratio Results.ctl) This control contains the pressure ratio calculations results.

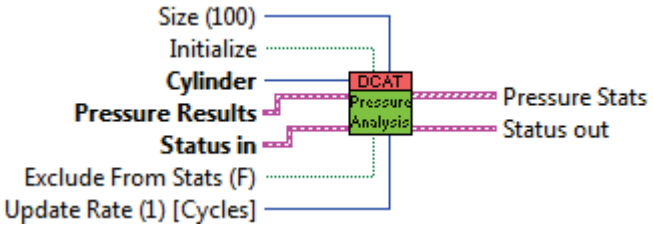
Size in (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Update Rate (1) [Cycles] (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

Pressure Stats RT (reentrant, normal priority) This VI calculates the statistics for the calculated pressure analysis values. This VI is good for up to 16 cylinders.



Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Pressure Results (Cluster, Pressure Results.ctl) This control contains the basic pressure metrics calculations results.

Pressure Stats (Array, Statistics Array.ctl) This array control contains the statistical cluster. Each element in the array correlates to a calculated value.

> **Stats** (Cluster, Stats.ctl) This control contains calculated statistical results.

Size (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

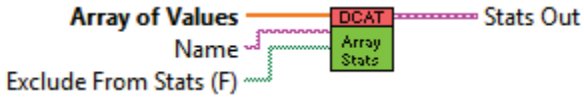
Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Update Rate (1) [Cycles] (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

Tools

Array Stats (reentrant, subroutine) This VI calculates statistics for an array of values. An optional array of Booleans can be used to exclude certain values from the statistics.



Array of Values (Array) This control contains the input values.

Value (Single Float) Parameter Value

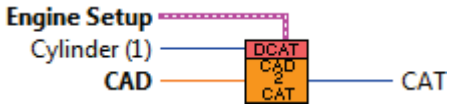
Exclude From Stats (F) (Array, Exclude From Stats Array.ctl) This control contains an array that can be used to exclude individual values from the statistics calculations. If the array is empty, all values will be used.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Name (String) This control defines the name of the parameter that the statistics are calculated for.

Stats Out (Cluster, Stats.ctl) This control contains calculated statistical results.

CAD2CAT_dcat (reentrant, subroutine) This VI calculates the crank angle degree (CAD) associated with a specific crank angle tick (CAT) as DCAT sees it.



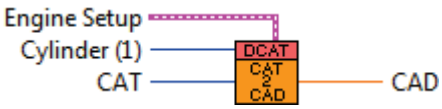
CAD (Single Float) Crank Angle Degrees

CAT (U16) Crank Angle Ticks

Cylinder (1) (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

CAT2CAD_dcat (reentrant, subroutine) This VI calculates the crank angle ticks (CAT) associated with a specific crank angle degree (CAD) as DCAT sees it.



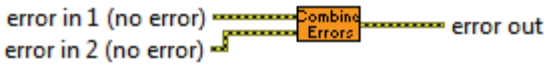
CAD (Single Float) Crank Angle Degrees

CAT (U16) Crank Angle Ticks

Cylinder (1) (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Combine Errors (reentrant, subroutine) This VI combines 2 error clusters. It is similar to the merge errors VI but it only merges 2 errors.



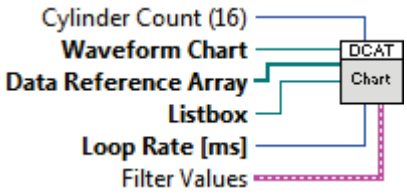
error in 1 (no error) (Cluster) This error cluster is combined with the other error cluster.

error in 2 (no error) (Cluster) This error cluster is combined with the other error cluster.

error out (Cluster) This error cluster is the result of combining the 2 input clusters.

Control Chart (reentrant, normal priority) This VI controls a chart on the host panel. The chart must be set up to take a 2D array of singles (wire a 2D array of singles into the array to change it and then disconnect it). This Uses a list box to allow the user to select 1 or more parameters in the "Data Reference Array" and plots them as a function of time (Selection Mode must be set to "1 or More Items" in the right click menu). The Filter Values input can be used to remove some parameters form the list box selection. The chart will clear when the selected values change. Trends with similar units will be placed on the same axis. If there are not enough axes for all the different units to be displayed, all units without an axis will be displayed on the first axis.

This VI needs to execute every loop cycle so that the table values update at the loop rate. The controls feeding them need to be outside of all tab controls so that they update regularly.



Cylinder Count (16) (U32) Numbers of cylinders to be calculated. This value must be greater than or equal to the number of cylinders calculated. Applying a number less than 16 allows the array sizes to be reduced to the actual size needed and operate more efficiently.

Data Reference Array (Array) This control is an array of references that contains the statistical data for the calculated values.

Array (Refnum) Array Control Reference

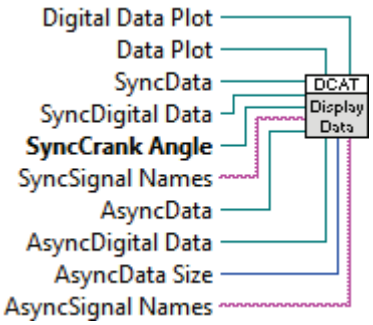
Filter Values (Cluster) This control filters the list of values in the listbox.

Listbox (Refnum) This control is a reference to the single column listbox that is used to select the properties to display on the waveform chart.

Loop Rate [ms] (U32) This control should be set to the target loop rate of the structure that contains the loop. The time axis is set using this control.

Waveform Chart (Refnum) This control is a reference to a waveform chart.

Control Data Plot (normal priority) This VI controls the display of the synchronous or asynchronous display for a single channel and all the measurements associated with it.



AsyncData (Refnum) This control contains the reference to the data.

AsyncData Size (I32) This control defines the original size of the asynchronous data. The actual size of the data array may be smaller due to the maximum size of the CalPoint.

AsyncDigital Data (Refnum) This control is a reference to the digital data.

AsyncSignal Names (String) This control contains the signal names separated by a comma.

Data Plot (Refnum) This control is a reference to the data plot x/y graph.

Digital Data Plot (Refnum) This control is a reference to the data plot x/y graph.

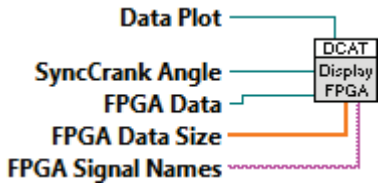
SyncCrank Angle (Refnum) This control is a reference to a crank angle array.

SyncData (Refnum) This control contains the reference to the data.

SyncDigital Data (Refnum) This control is a reference to the digital data.

SyncSignal Names (String) This control contains the signal names separated by a comma.

Control FPGA Plot (normal priority) This VI controls the display of the synchronous or asynchronous display for a single channel and all the measurements associated with it.



Data Plot (Refnum) This control is a reference to the data plot x/y graph.

FPGA Data (Refnum) This control contains the reference to the data.

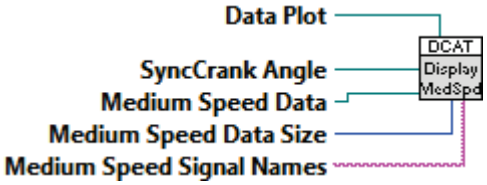
FPGA Data Size (Array) This control contains the size of the FPGA data streams.

FPGA Data Size (Single Float) FPGA Data Size

FPGA Signal Names (String) This control contains the signal names separated by a comma.

SyncCrank Angle (Refnum) This control is a reference to a crank angle array.

Control Medium Speed Plot (normal priority) This VI controls the display of the synchronous or asynchronous display for a single channel and all the measurements associated with it.



Data Plot (Refnum) This control is a reference to the data plot x/y graph.

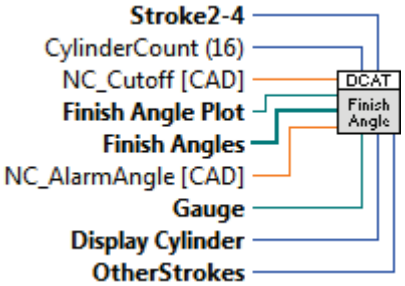
Medium Speed Data (Refnum) This control contains the reference to the data.

Medium Speed Data Size (I32) This control contains the size of the medium speed data.

Medium Speed Signal Names (String) This control contains the signal names separated by a comma.

SyncCrank Angle (Refnum) This control is a reference to a crank angle array.

Control NC Finish Angle Plot (reentrant, normal priority) This VI may be used to control a gauge indicator to that represents the finishing angle(s) of the next cycle calculations.



CylinderCount (16) (U32) Numbers of cylinders to be calculated. This value must be greater than or equal to the number of cylinders calculated. Applying a number less than 16 allows the array sizes to be reduced to the actual size needed and operate more efficiently.

Display Cylinder (I32) This control specifies the cylinder to display on the plot.

Finish Angle Plot (Refnum) This control contains the reference to the plot of the finish angle.

Finish Angles (Array) This control contains the finishing angles to be plotted.

Finish Angle (Refnum) Finishing Angles

Gauge (Refnum) This control contains the reference to a gauge display to show the finish angles.

NC_AlarmAngle [CAD] (Single Float) This control defines the angle at which the calculations must finish.

NC_Cutoff [CAD] (Single Float) This control contains the cutoff angle for the next cycle calculations.

OtherStrokes (U8) This control defines the start of the window used to calculate the polytrophic coefficient for compression.

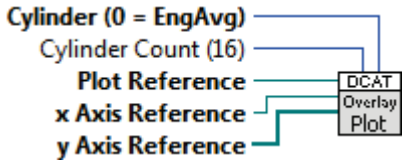
Stroke2-4 (Enum U32) Specifies a 2 or 4 stroke engine.

0: 4-Stroke: 2 engine revolution per cycle

1: 2-Stroke: 1 engine revolution per cycle

2: Other: User specified revolutions per cycle

Control Overlay Plot (reentrant, normal priority) This VI controls a typical plot on the host VI. The VI accepts a plot XY Graph reference, a 1D array reference for the x axis, and a 2D array for the y axis. The axis labels are automatically set up based on the array names and units. Up 16 cylinders of data will be plotted based on the value of Cylinder Count.



Cylinder (0 = EngAvg) (U32) Numbers of cylinders to be calculated. This value must be greater than or equal to the number of cylinders calculated. Applying a number less than 16 allows the array sizes to be reduced to the actual size needed and operate more efficiently.

Cylinder Count (16) (U32) Numbers of cylinders to be calculated. This value must be greater than or equal to the number of cylinders calculated. Applying a number less than 16 allows the array sizes to be reduced to the actual size needed and operate more efficiently.

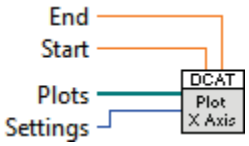
Plot Reference (Refnum) This control is a reference to the X/Y graph of the data.

x Axis Reference (Refnum) This control is a reference to the x-axis array.

y Axis Reference (Array) This control is a reference to the data array.

y Axis Reference (Refnum) Y Axis Reference

Control Plot X Axis (normal priority) This VI may be used to control the X-axis scaling of a large number of XY graphs that use crank angle as the X-axis.



End (Single Float) This control defines the end angle when using custom settings.

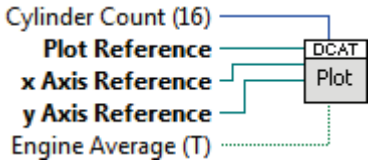
Plots (Array) This control contains the references to the plots to be controlled.

PR MFB Plot (Refnum) Plot

Settings (Enum U32) This control can be set to different pre-set windows or set to custom.

Start (Single Float) This control defines the start angle when using custom settings.

Control Plot (reentrant, normal priority) This VI controls a typical plot on the host VI. The VI accepts a plot XY Graph reference, a 1D array reference for the x axis, and a 2D array for the y axis. The axis labels are automatically set up based on the array names and units. Up 16 cylinders of data will be plotted based on the value of Cylinder Count.



Cylinder Count (16) (U32) Numbers of cylinders to be calculated. This value must be greater than or equal to the number of cylinders calculated. Applying a number less than 16 allows the array sizes to be reduced to the actual size needed and operate more efficiently.

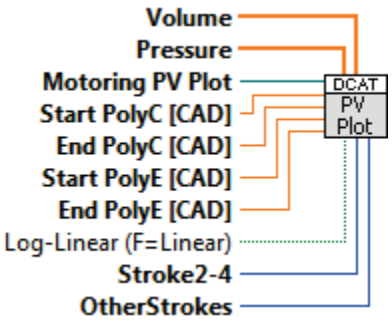
Engine Average (T) (Boolean) When true, this control adds an engine average to the plotted data.

Plot Reference (Refnum) This control is a reference to the X/Y graph of the data.

x Axis Reference (Refnum) This control is a reference to the x-axis array.

y Axis Reference (Refnum) This control is a reference to the data array.

Control PV Plot (reentrant, normal priority) This VI is used to control the PV diagram plot.



End PolyC [CAD] (Single Float) This control defines the end of the window used to calculate the polytrophic coefficient for compression.

End PolyE [CAD] (Single Float) This control defines the end of the window used to calculate the polytrophic coefficient for expansion.

Log-Linear (F=Linear) (Boolean) This control determines if the X/Y graph uses linear or log scales.

Motoring PV Plot (Refnum) This control is a reference to the X/Y graph.

OtherStrokes (U8) This control defines the start of the window used to calculate the polytrophic coefficient for compression.

Pressure (Array) This control contains the pressure data.

Pressure (Single Float) Pressure

Start PolyC [CAD] (Single Float) This control defines the start of the window used to calculate the polytrophic coefficient for compression.

Start PolyE [CAD] (Single Float) This control defines the start of the window used to calculate the polytrophic coefficient for expansion.

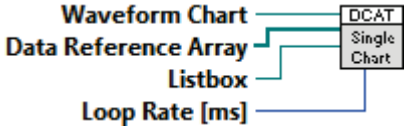
Stroke2-4 (Enum U32) This control defines if the engine is a 2 or 4 stroke per cycle.

Volume (Array) This control contains the volume array.

Volume (Single Float) Volume

Control Single Chart (reentrant, normal priority) This VI controls a chart on the host panel. The chart must be set up to take a singles. This VI uses a list box to allow the user to select 1 or more parameters in the "Data Reference Array" and plots them as a function of time (Selection Mode must be set to "1 or More Items" in the right click menu). The chart will clear when the selected values change. Trends with similar units will be placed on the same axis. If there are not enough axes for all the different units to be displayed, all units without an axis will be displayed on the first axis.

This VI needs to execute every loop cycle so that the table values update at the loop rate. The controls feeding them need to be outside of all tab controls so that they update regularly.



Data Reference Array (Array) This control is an array of references that contains the statistical data for the calculated values.

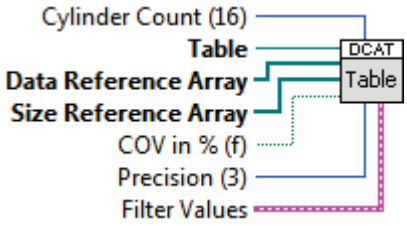
Data Reference (Refnum) Data Reference

Listbox (Refnum) This control is a reference to the single column listbox that is used to select the properties to display on the waveform chart.

Loop Rate [ms] (U32) This control should be set to the target loop rate of the structure that contains the loop. The time axis is set using this control.

Waveform Chart (Refnum) This control is a reference to a waveform chart.

Control Table (reentrant, normal priority) This VI is designed to run in the host application as a way to format and display tables. The "Data Reference Array" contains references to 2d arrays of statistical values (16 x 5). The first dimension in the array represents the cylinder and the second dimension represents the statistic (Current Value, Mean, Max, Min, SD). All values in the array have the same units. A second array of references, "Size Reference Array", must match the size of the "Data Reference Array". This 1d array has 16 values that represent the number of values used to create statistics. Each value corresponds to a cylinder. The "Table" reference defines which table will be updated with the statistics. "Enable Values" turns on or off the display of any of the statistics for all the parameters.



COV in % (f) (Boolean) This control is used to convert the coefficient of variance to a percent instead of a fraction.

Cylinder Count (16) (U32) Numbers of cylinders to be calculated. This value must be greater than or equal to the number of cylinders calculated. Applying a number less than 16 allows the array sizes to be reduced to the actual size needed and operate more efficiently.

Data Reference Array (Array) This control is an array of references to the statistical results to be displayed on the table.

Data Array (Refnum) Data Array

Filter Values (Cluster) This control filters the list of values in the listbox.

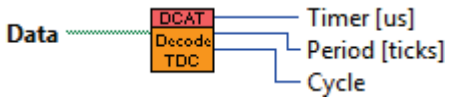
Precision (3) (I16) This control sets the precision of the data displayed in the table.

Size Reference Array (Array) This control is an array of references that contain the data set size information. Each element in the array corresponds to each element in the data reference array.

Size Array (Refnum) Size Array

Table (Refnum) This control is a reference to the table display of the data.

Decode TDC (reentrant, subroutine) This VI decodes the TDC signal for the Asynchronous channels to provide the cycle number and the Speed. The TDC signal must in phase with cylinder 1.



Cycle (U32, Cycle.ctl) Cylinder associated with the current data set. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Data (Array) This array control contains the serialized data stream that is to be decoded into a cycle number, cycle period, and FPGA timer.

Bit (Boolean) A single bit in the data stream.

Period [ticks] (U32, Period.ctl) Period of a single engine rotation. It can be inverted to determine the engine speed.

Timer [us] (U32, Timer.ctl) Timer contains the current value of the FPGA free timer. The time is updated once every cycle.

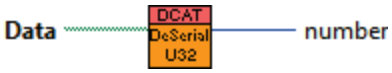
DeSerializeU16 (reentrant, subroutine) This VI de-serializes a U16 number from the DCAT digital data.

Data (Array) This array control contains the serialized data stream that is to be decoded into a cycle number, cycle period, and FPGA timer.

Bit (Boolean) A single bit in the data stream.

Number (U16) This control is the de-serialized U16 number.

DeSerializeU32 (reentrant, subroutine) This control is the de-serialized U8 number.



Data (Array) This array control contains the serialized data stream that is to be decoded into a cycle number, cycle period, and FPGA timer.

Bit (Boolean) A single bit in the data stream.

Number (U32)

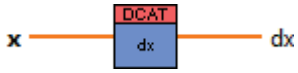
DeSerializeU8 (reentrant, subroutine) This VI de-serializes a U8 number from the DCAT digital data.

Data (Array) This array control contains the serialized data stream that is to be decoded into a cycle number, cycle period, and FPGA timer.

Bit (Boolean) A single bit in the data stream.

Number (U8) This control is the de-serialized U8 number.

dx (reentrant, subroutine) This VI calculates dx provided x. The differential values are using a second order center difference with the boundary conditions as the initial and final conditions.



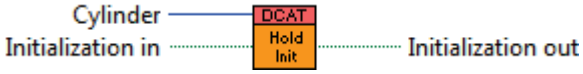
dx (Array) This control is a the results of the differential array calculations.

dx Element (Single Float) dx Element

x (Array) This control is the input of the differential array calculations.

x Element (Single Float) x Element

Hold Init (reentrant, subroutine) This VI holds the initialization flag until each cylinder has use the flag to clear the appropriate data. When a value of true enters initialize in, a set of 16 flags are set. As the cylinders execute the output initialization is then true for each cylinder that has not run after initialize in was true.

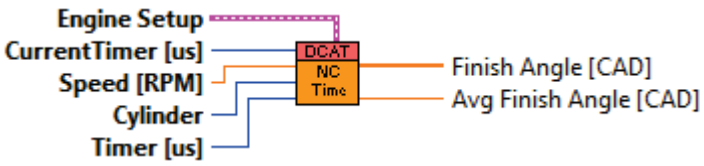


Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Initialize in (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Initialize out (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Next Cycle Time Delay (reentrant, subroutine) This VI calculates the estimated finish time of a calculations in degrees past TDC of the cycle. This VI is intended to give some estimate of the delay in calculations for the next cycle control. This VI will keep the most recent cycle times for up to 16 cylinders.



Avg Finish Angle [CAD] (Single Float)

CurrentTimer [us] (U32) This control represents the current FPGA free timer value.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

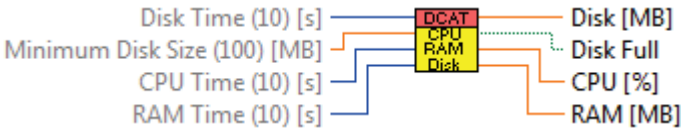
Finish Angle [CAD] (Array) This control represents the estimated angle past TDC associated with the current timer value. Each value in the array corresponds to a different cylinder.

Finishing Angle [CAD] (Single Float) Finishing Angle [CAD]

Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle.

Timer [us] (U32, Timer.ctl) Timer contains the current value of the FPGA free timer. The time is updated once every cycle.

Real-Time Resources (normal priority) This VI periodically checks the CPU, RAM, and Disk Space when available.



CPU Time (10) [s] (U32) This control defines how often to update the CPU usage.

CPU [%] (Single Float) This control displays the CPU usage.

Disk Full (Boolean) This control is true when the disk is full.

Disk Time (10) [s] (U32) This control defines how often to update the disk space.

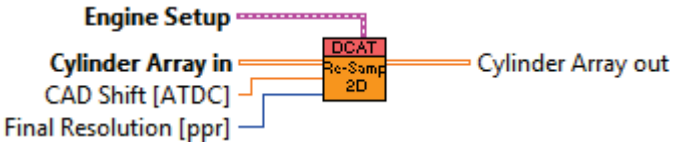
Disk [MB] (Single Float) This control displays the free disk space.

Minimum Disk Size (100) [MB] (Single Float) This control defines the minimum disk space before the disk is considered full.

RAM Time (10) [s] (U32) This control defines how often to update the RAM use.

RAM [MB] (Single Float) This control displays the free RAM.

Resample Cycle 2D (reentrant, subroutine) This VI re-samples the cycle array by first shifting the array by the appropriate number of indexes and then by interpolating between indexes. If ether step has a zero offset, the step is skipped.



CAD Shift [ATDC] (Single Float) This control defines the angle that the data is to be shifted. Shifting the data by a integer number of samples will result in a faster calculation than shifting the array by a fractional number of samples.

Cycle Array in (Array) This control is an original array.

Array Element (Single Float) Array Element

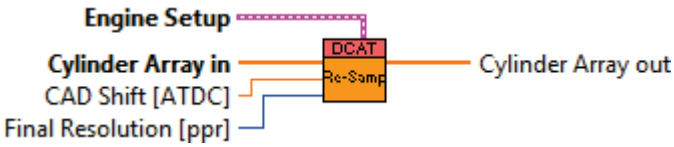
Cycle Array out (Array) This control contains the resulting array.

Array Element (Single Float) Array Element

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Final Resolution [ppr] (I32) This control defines the final resolution of the output array. A value of zero will cause the output to be the same as the input.

Resample Cycle (reentrant, subroutine) This VI re-samples the cycle array by first shifting the array by the appropriate number of indexes and then by interpolating between indexes. If ether step has a zero offset, the step is skipped.



CAD Shift [ATDC] (Single Float) This control defines the angle that the data is to be shifted. Shifting the data by a integer number of samples will result in a faster calculation than shifting the array by a fractional number of samples.

Cycle Array in (Array) This control is an original array.

Array Element (Single Float) Array Element

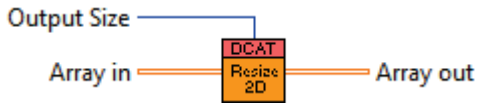
Cycle Array out (Array) This control contains the resulting array.

Array Element (Single Float) Array Element

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Final Resolution [ppr] (I32) This control defines the final resolution of the output array. A value of zero will cause the output to be the same as the input.

Resize Array 2D (reentrant, subroutine) This VI converts an array to an array of any size. The input array is interpolated to determine the output array. If the input array is empty, an array of zeros is used. If the input array size matches the output array size, the input array is passed directly to the output.



Array in (Array) The control contains the input array data.

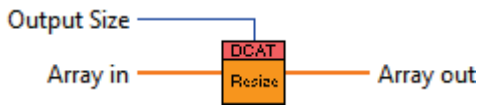
Array Element (Single Float) Array Element

Array out (Array) The control contains the input array data.

Array Element (Single Float) Array Element

Output Size (I32) This control defines the array output size.

Resize Array (reentrant, subroutine) This VI converts an array to an array of any size. The input array is interpolated to determine the output array. If the input array is empty, an array of zeros is used. If the input array size matches the output array size, the input array is passed directly to the output.



Array in (Array) The control contains the input array data.

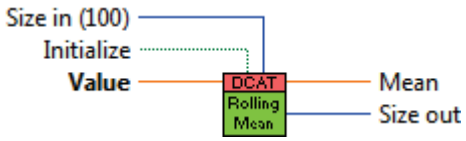
Array Element (Single Float) Array Element

Array out (Array) The control contains the input array data.

Array Element (Single Float) Array Element

Output Size (I32) This control defines the array output size.

Rolling Mean (reentrant, subroutine) This VI keeps a rolling average with a buffer size equal to the Size in control. Statistics are calculated on the start of the buffer until it fills and then on the entire buffer. This VI stores a history of data and each instance must only be used for a single measurement.



Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

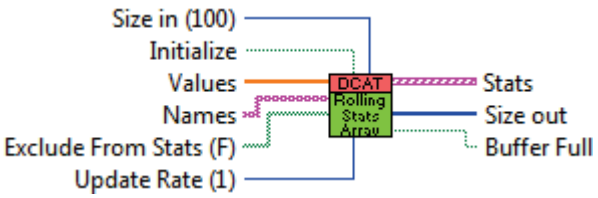
Mean (Single Float) This control is the mean of the data set.

Size in (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Size out (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Value (Single Float) This control is the current value.

Rolling Stats Array (reentrant, subroutine) This VI keeps rolling statistics with a buffer size equal to the Size in control. Statistics are calculated on the start of the buffer until it fills and then on the entire buffer. This VI stores a history of data and each instance must only be used for a single measurement.



Buffer Full (Boolean) This control is true when the rolling buffer is full.

Exclude From Stats (F) (Array, Exclude From Stats Array.ctl) This control contains an array that can be used to exclude individual values from the statistics calculations. If the array is empty, all values will be used.

> **Exclude From Stats (F)** (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Names (Array) This control defines the names of the property. Each element in the name array corresponds to a statistics cluster, size out, exclude from stats, size out, and value.

> **Name** (String) Property Name

Size in (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Size out (Array) This control reports the actual sample sizes used in the statistical calculations. Each element in the size out array corresponds to a statistics cluster, size out, exclude from stats, name, and value.

> **Size out** (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Stats (Array) This control contains all the results for the statistical calculations. Each element in the stats array corresponds to a name, size out, exclude from stats, size out, and value.

> **Stats** (Cluster, Stats.ctl) This control contains calculated statistical results.

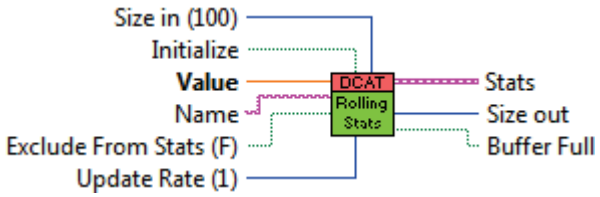
Update Rate (1) (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

Values (Array) This control defines the current value for the properties. Each element in the value array corresponds to a statistics cluster, size out, exclude from stats, size out, and name.

> **Value** (Single Float) Value

Rolling Stats (reentrant, subroutine) This VI keeps rolling statistics with a buffer size equal to the Size in control. Statistics are calculated on the start of

the buffer until it fills and then on the entire buffer. This VI stores a history of data and each instance must only be used for a single measurement.



Buffer Full (Boolean) This control indicates that the buffers have filled.

Exclude From Stats (F) (Boolean, Exclude From Stats.ctl) This control is set true if the current value is to be excluded from the statistical calculations. This may be desirable if statistics are required for a subset sampling population based on some criteria. Example: if you need to determine the average peak pressure for all cylinder events that are not considered a misfire.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Name (String) This control defines the name of the parameter.

Size in (100) (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Size out (I32, Size.ctl) This control defines the size of rolling average used in statistics.

Stats (Cluster, Stats.ctl) This control contains calculated statistical results.

Update Rate (1) (I32, Update Rate.ctl) Update rate defines the number of cycle between rolling statistical calculations. The minimum, maximum, and mean values are updated every cycle.

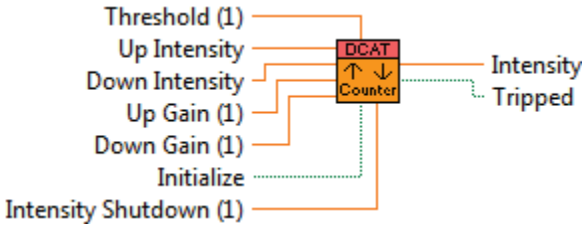
Value (Single Float) This control is the current value.

Timer Lite (reentrant, subroutine) This VI sets the output true for a single loop after the timer expires. The timer automatically resets.

Time Target [ms] (U32) This control determines the timer length.

Time has Elapsed (Boolean) This control indicates that the time has elapsed.

Up Down Counter (reentrant, subroutine) This VI is an up-down counter.



Down Gain (1) (Single Float) This control defines the gain on the down intensity.

Down Intensity (Single Float) This control defines the decrease in intensity each iteration.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Intensity (Single Float) This is the resulting intensity value.

Intensity Shutdown (1) (Single Float)

Threshold (1) (Single Float) This control determines the threshold where the intensity is great enough to set a true value.

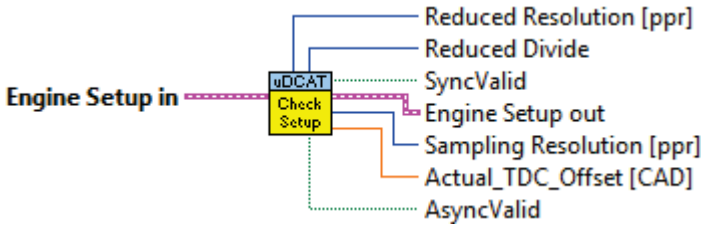
Tripped (Boolean) This control is true when the intensity is greater than the threshold.

Up Gain (1) (Single Float) This control defines the gain on the up intensity.

Up Intensity (Single Float) This control defines the increase in the intensity each iteration.

uDCAT

Check Setup (normal priority) This VI checks the engine setup and the synchronous measurements setup to ensure that the encoder is properly setup for uDCAT.



Actual TDC Offset [CAD] (Single Float) This control displays the actual top dead center offset that will be used by the data acquisition system. It is the closest integer sample offset to the desired top dead center offset.

Asynchronous Valid (Boolean) This control indicates true when the synchronous sampling setup is acceptable. It returns false when an error condition exists and overwrites the settings the default settings.

Engine Setup in (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Engine Setup out (Cluster, Engine Setup.ctl) This control contains general engine setup information.

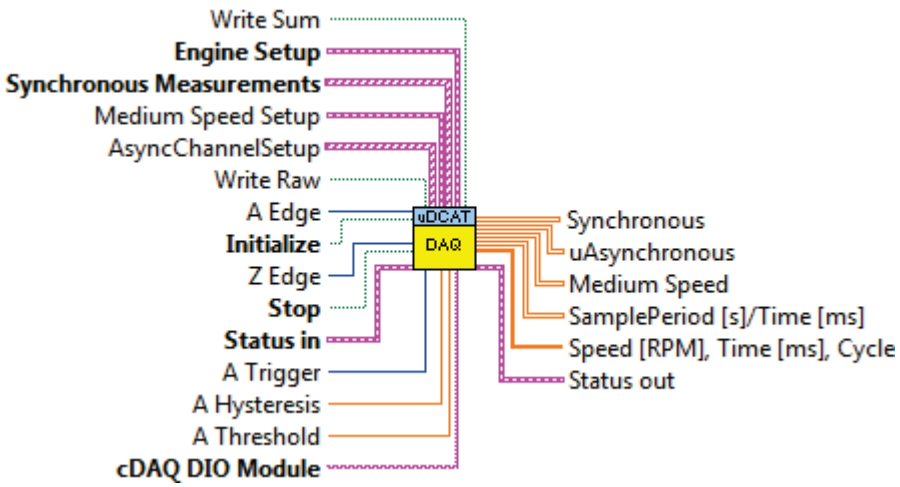
Reduced Divide (U32) This control displays the divide value required to achieve the reduced resolution.

Reduced Resolution [ppr] (U32) This control displays the reduced resolution encoder signal that is recorded with the asynchronous sampling to ensure that the encoder signal can be interpreted properly.

Sampling Resolution [ppr] (U32) This control displays the final sample resolution to be used by the data acquisition system.

Synchronous Valid (Boolean) This control indicates true when the asynchronous sampling setup is acceptable. It returns false when an error condition exists and overwrites the settings the default settings.

Read Channels (normal priority)



A Edge (Enum U16) This control is used to select between a rising or a falling edge to trigger the A signal.

A Hysteresis (Single Float) This control sets the hysteresis level when using the APFI channel for the A signal.

A Threshold (Single Float) This control sets the threshold level when using the APFI channel for the A signal.

A Trigger (Enum U32) This control determines is the A signal is on the PFI 0 or the APFI channel.

Asynchronous Channels (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

> **Measurement Setup** (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

> **Medium Speed Sample** (Double Float) Medium Speed Sample

Medium Speed Setup (Cluster, Medium Speed Setup.ctl) This control contains the medium speed DAQ setup.

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

- **SamplePerid [s] / Time [ms]** (Double Float) Sample Period [s] / Time [ms]

Speed [RPM], Time [ms], Cycle (Array, STC.ctl) This array control contains the speed, time, and cycle. The 0 index in the array is always speed. The 1 index is always time. The 2 index is always cycle number. This array is only used by uDCAT.

- **Value** (Double Float) Speed, Time, or Cycle

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Stop (Boolean, Stop.ctl) This control stops the VI and clears the stored data in the VI. It should be used during the last loop of the cycle.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Synchronous Sample

Synchronous Measurements (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

- **Measurement Setup** (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Write Raw (Boolean) This control will set an output true when saving a raw data file.

Write Sum (Boolean) This control will set an output true when saving a summary data file.

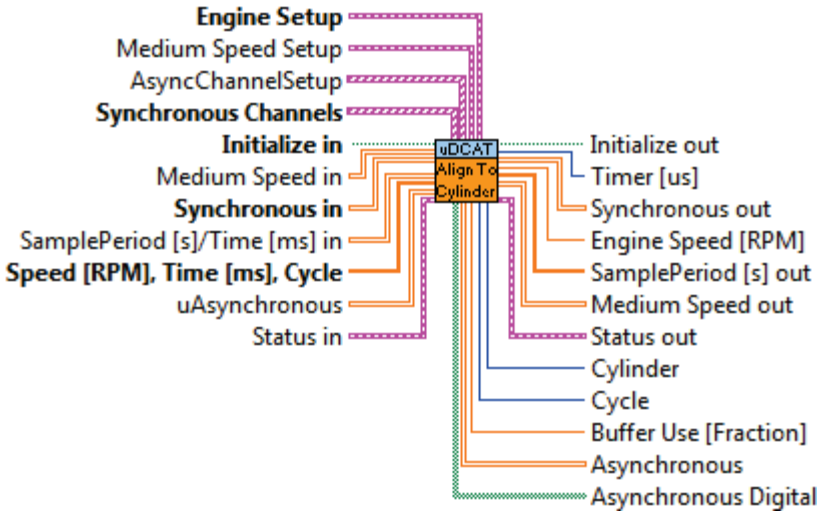
Z Edge (Enum U16) This control is used to select between a rising or a falling edge to trigger the Z signal.

cDAQ DIO Module (String) This control defines the name of the module used to read the encoder signals. It is only valid when using a cDAQ chassis. The module name is found in the measurements and automation explorer.

uAsynchronous (Array, uAsynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Double Float) Asynchronous Sample

uAlign to Cylinder (normal priority) This VI takes the raw data streams and divides it up into individual cylinder events. This VI is to be called repeatedly and faster than each cylinder event. It will return a single cylinder's data when the complete cylinder arrays are available. If no new cylinder information is available, a cylinder number of -1 is returned. The VI will return an error if all the information cannot be processed but the VI will attempt to recover.



Asynchronous (Array, Asynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Single Float) Asynchronous Sample

Asynchronous Channels (Array, Asynchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the asynchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Asynchronous Digital (Array, Asynchronous Digital.ctl) Asynchronous Digital signals recorded by the DAQ system. The first dimension corresponds to each channel of digital data. The second dimension corresponds to each sample.

Digital Sample (Boolean) Digital Sample

Buffer Use [Fraction] (Single Float, Buffer Use.ctl) This control displays the fraction of the buffer or queue that is currently used.

Cycle (U32, Cycle.ctl) Cylinder associated with the current data set. A cylinder number of -1 or 0 is used when cycle data is not avaliable. Valid cylinder number is between 1 and 16.

Cylinder (I32, Cylinder.ctl) Cylinder associated with the current data. A cylinder number of -1 or 0 is used when cycle data is not available. Valid cylinder number is between 1 and 16.

Engine Setup (Cluster, Engine Setup.ctl) This control contains general engine setup information.

Engine Speed [RPM] (Single Float, Engine Speed.ctl) Engine Speed for the current cycle of data. It is updated once every cycle. This control is defined by "Engine Speed.ctl".

Initialize in (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Initialize out (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed Setup (Cluster, Medium Speed Setup.ctl) This control contains the medium speed DAQ setup.

Medium Speed in (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

Medium Speed out (Array, Medium Speed sgl.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Medium Speed Sample

SamplePeriod [s] out (Array, SamplePeriod.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

Waveform (Single Float) Sample Period

SamplePeriod [s]/Time [ms] in (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

Speed [RPM], Time [ms], Cycle (Array, STC.ctl) This array control contains the speed, time, and cycle. The 0 index in the array is always speed. The 1 index is always time. The 2 index is always cycle number. This array is only used by uDCAT.

Value (Double Float) Speed, Time, or Cycle

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Synchronous Channels (Array, Synchronous Measurements.ctl) This control is an array of measurement setup clusters defining all the synchronous measurements. Each element in the array represents a single measurement. The name, type, scaling, associated cylinders, and other properties are defined for each measurement.

Measurement Setup (Cluster, Measurement.ctl) This control defines the properties associated with a measurement.

Synchronous in (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample. This control is defined by "Synchronous.ctl".

Synchronous Sample (Single Float) Synchronous Sample

Synchronous out (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample. This control is defined by "Synchronous.ctl".

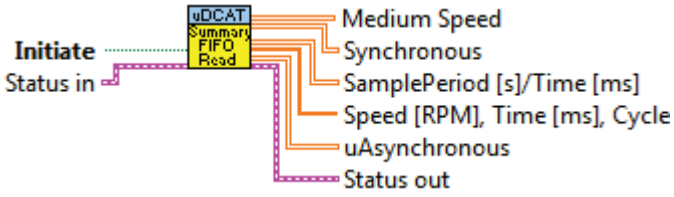
Synchronous Sample (Single Float) Synchronous Sample

Timer [us] (U32, Timer.ctl) Timer contains the current value of the FPGA free timer. The time is updated once every cycle. This control is defined by "Timer.ctl".

uAsynchronous (Array, uAsynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Double Float) Asynchronous Sample

uRT FIFO Read (normal priority) This VI reads the queue of raw data for the real time processing VIs.



Initiate (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

Speed [RPM], Time [ms], Cycle (Array, STC.ctl) This array control contains the speed, time, and cycle. The 0 index in the array is always speed. The 1 index is always time. The 2 index is always cycle number. This array is only used by uDCAT.

Value (Double Float) Speed, Time, or Cycle

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

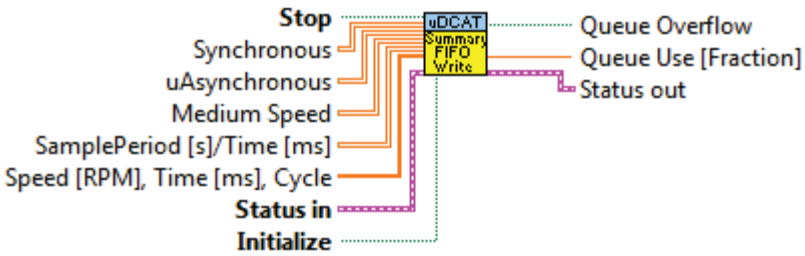
Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

uAsynchronous (Array, uAsynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Double Float) Asynchronous Sample

uRT FIFO Write (normal priority) This VI takes the raw data and writes it to a queue that can be read by the streaming to disk VIs.



Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted. This control is defined by "Initialize.ctl".

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

Queue Overflow (Boolean, Queue Overflow.ctl) This control displays true if any of the queues are full.

Queue Use [Fraction] (Single Float, Buffer Use.ctl) This control displays the fraction of the buffer or queue that is currently used.

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

Speed [RPM], Time [ms], Cycle (Array, STC.ctl) This array control contains the speed, time, and cycle. The 0 index in the array is always speed. The 1 index is always time. The 2 index is always cycle number. This array is only used by uDCAT.

Value (Double Float) Speed, Time, or Cycle

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Stop (Boolean, Stop.ctl) This control stops the VI and clears the stored data in the VI. It should be used during the last loop of the cycle.

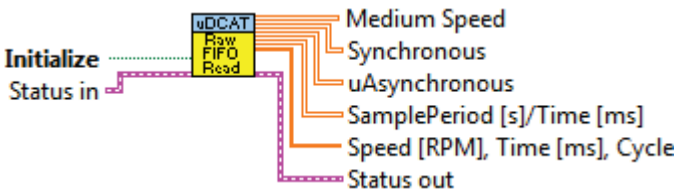
Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

uAsynchronous (Array, uAsynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Double Float) Asynchronous Sample

uST FIFO Read (normal priority) This VI reads the queue of raw data for the real time processing VIs.



Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the

timestamp of the samples. The second dimension in the array corresponds to each sample.

Medium Speed Sample (Double Float) Medium Speed Sample

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

SamplePerid [s] / Time [ms] (Double Float) Sample Period [s] / Time [ms]

Speed [RPM], Time [ms], Cycle (Array, STC.ctl) This array control contains the speed, time, and cycle. The 0 index in the array is always speed. The 1 index is always time. The 2 index is always cycle number. This array is only used by uDCAT.

Value (Double Float) Speed, Time, or Cycle

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

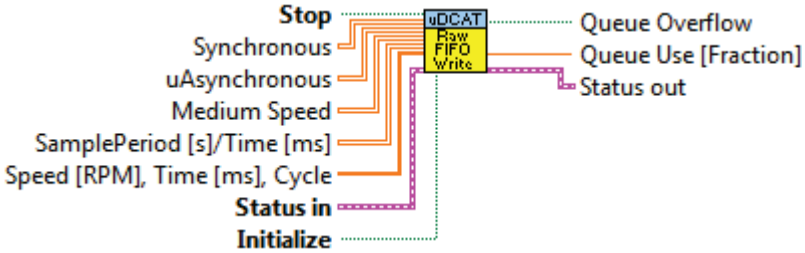
Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Synchronous Sample (Single Float) Synchronous Sample

uAsynchronous (Array, uAsynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Double Float) Asynchronous Sample

uST FIFO Write (normal priority) This VI takes the raw data and writes it to a queue that can be read by the streaming to disk VIs.



Initialize (Boolean, Initialize.ctl) This control is used to reset data stored in the VI. This should be set on the first iteration of a loop. It should also run when a loop is restarted.

Medium Speed (Array, Medium Speed.ctl) Medium Speed array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The first measurement is the timestamp of the samples. The second dimension in the array corresponds to each sample.

- **Medium Speed Sample** (Double Float) Medium Speed Sample

Queue Overflow (Boolean, Queue Overflow.ctl) This control displays true if any of the queues are full.

Queue Use [Fraction] (Single Float, Buffer Use.ctl) This control displays the fraction of the buffer or queue that is currently used.

SamplePeriod [s]/Time [ms] (Array, SamplePeriod and Time.ctl) Synchronous array contains the sample period from the previous sample to the current sample. Each element in the array corresponds to an element in the data arrays.

- **SamplePerid [s] / Time [ms]** (Double Float) Sample Period [s] / Time [ms]

Speed [RPM], Time [ms], Cycle (Array, STC.ctl) This array control contains the speed, time, and cycle. The 0 index in the array is always speed. The 1 index is always time. The 2 index is always cycle number. This array is only used by uDCAT.

- **Value** (Double Float) Speed, Time, or Cycle

Status in (Cluster, Status.ctl) This control contains the status of the real time process.

Status out (Cluster, Status.ctl) This control contains the status of the real time process.

Stop (Boolean, Stop.ctl) This control stops the VI and clears the stored data in the VI. It should be used during the last loop of the cycle.

Synchronous (Array, Synchronous.ctl) Synchronous array contains the samples recorded by the data acquisition system. The first dimension in the array corresponds to each measurement. The second dimension in the array corresponds to each sample.

- **Synchronous Sample** (Single Float) Synchronous Sample

uAsynchronous (Array, uAsynchronous.ctl) Asynchronous array contains the samples recorded by the data acquisition system. The first dimension in the

array corresponds to each measurement. The second dimension in the array corresponds to each sample.

Asynchronous Sample (Double Float) Asynchronous Sample

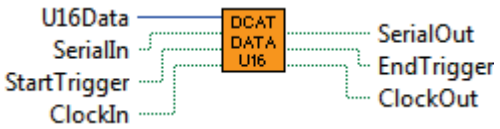
FGPA

dcat_data16_reva (reentrant, normal priority) This VI accepts a single U16 input value and serializes it with a UART protocol to be sampled by an S-Series digital input. Multiple dcat_data16 VIs can be daisy-chained to provide multiple serialized U16 values. This data is sampled along with angle-based and time-based data and decoded by real-time DCAT VIs. This data is stored with the analog data. This enables various control and DAQ parameters to be known for each engine cycle.

Each 16-bit word is encoded with 1 start bit (logic high) and 2 stop bits (logic low). The rising edge of the first start bit coincides with the rising edge of the first sample clock after the start trigger. Each data bit is encoded with 2 sample clocks. It is required by the user to ensure that all data bits can fit within the number of sample clocks within each engine cycle. Otherwise the decoding logic will be out of sync.

The first VI in the daisy-chain should have a constant FALSE boolean wired to the SerialIn input. The EncABOut and EncZOut outputs of the DCAT Sample Supervisor should be wired to the ClockIn and StartTrigger, respectively. The SerialOut, EndTrigger and ClockOut outputs should be wired directly to the SerialIn, StartTrigger and ClockIn of the next daisy-chained VI.

Only one data type may be serialized to one data stream.



ClockIn (Boolean) Sample Clock In

ClockOut (Boolean) Sample Clock Out

EndTrigger (Boolean) Start Trigger Out

SerialIn (Boolean) Serial Data Line In

SerialOut (Boolean) Serial Data Line Out

StartTrigger (Boolean) Start Trigger In

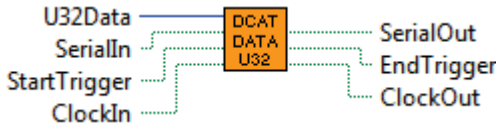
U16Data (U16) This control contains the data to be encoded.

dcat_data32_reva (reentrant, normal priority) This VI accepts a single U32 input value and serializes it with a UART protocol to be sampled by an S-Series digital input. Multiple dcat_data32 VIs can be daisy-chained to provide multiple serialized U32 values. This data is sampled along with angle-based and time-based data and decoded by real-time DCAT VIs. This data is stored with the analog data. This enables various control and DAQ parameters to be known for each engine cycle.

Each 32-bit word is encoded with 1 start bit (logic high) and 2 stop bits (logic low). The rising edge of the first start bit coincides with the rising edge of the first sample clock after the start trigger. Each data bit is encoded with 2 sample clocks. It is required by the user to ensure that all data bits can fit within the number of sample clocks within each engine cycle. Otherwise the decoding logic will be out of sync.

The first VI in the daisy-chain should have a constant FALSE boolean wired to the SerialIn input. The EncABOut and EncZOut outputs of the DCAT Sample Supervisor should be wired to the ClockIn and StartTrigger, respectively. The SerialOut, EndTrigger and ClockOut outputs should be wired directly to the SerialIn, StartTrigger and ClockIn of the next daisy-chained VI.

Only one data type may be serialized to one data stream.



ClockIn (Boolean) Sample Clock In

ClockOut (Boolean) Sample Clock Out

EndTrigger (Boolean) Start Trigger Out

SerialIn (Boolean) Serial Data Line In

SerialOut (Boolean) Serial Data Line Out

StartTrigger (Boolean) Start Trigger In

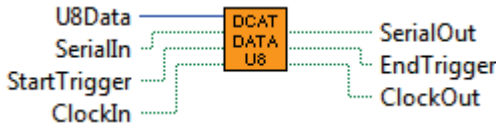
U32Data (U32) This control contains the data to be encoded.

dcat_data8_reva (reentrant, normal priority) This VI accepts a single U8 input value and serializes it with a UART protocol to be sampled by an S-Series digital input. Multiple dcat_data8 VIs can be daisy-chained to provide multiple serialized U8 values. This data is sampled along with angle-based and time-based data and decoded by real-time DCAT VIs. This data is stored with the analog data. This enables various control and DAQ parameters to be known for each engine cycle.

Each 8-bit word is encoded with 1 start bit (logic high) and 2 stop bits (logic low). The rising edge of the first start bit coincides with the rising edge of the first sample clock after the start trigger. Each data bit is encoded with 2 sample clocks. It is required by the user to ensure that all data bits can fit within the number of sample clocks within each engine cycle. Otherwise the decoding logic will be out of sync.

The first VI in the daisy-chain should have a constant FALSE boolean wired to the SerialIn input. The EncABOut and EncZOut outputs of the DCAT Sample Supervisor should be wired to the ClockIn and StartTrigger, respectively. The SerialOut, EndTrigger and ClockOut outputs should be wired directly to the SerialIn, StartTrigger and ClockIn of the next daisy-chained VI.

Only one data type may be serialized to one data stream.



ClockIn (Boolean) Sample Clock In

ClockOut (Boolean) Sample Clock Out

EndTrigger (Boolean) End Trigger Out

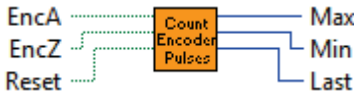
SerialIn (Boolean) Serial Data Line In

SerialOut (Boolean) Serial Data Line Out

StartTrigger (Boolean) Start Trigger In

U8Data (U8) This control contains the data to be encoded.

dcat_enc_count (reentrant, subroutine) This VI counts number of encoder A pulses per encoder Z pulse. It reports the last calculated value and the min/max counts. The min, max, and last counts should always be equal for the system to function properly. The reset sets the min and max values.



EncA (Boolean) Encocder A signal.

EncZ (Boolean) Encoder Z signal.

Last (U16) Displays the last encoder counter.

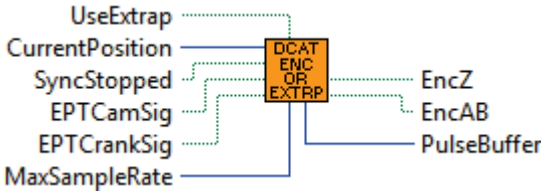
Max (U16) Displays the maximum encoder count.

Min (U16) Displays the minimum encoder count.

Reset (Boolean) Resets the counters.

dcat_enc_or_extrap (reentrant, normal priority) When UseExtrap is FALSE, the Crank and Cam signals from the EPT are connected to the EncAB and EncZ outputs, respectively.

When UseExtrap is TRUE, the least significant bit of CurrentPosition is used as a higher resolution encoder signal connected to EncAB. EncZ is high when CurrentPosition is zero. This mode can provide a higher resolution sample clock to the DAQ cards, using the extrapolated crank angle ticks of the EPT. However, the extrapolated crank angle ticks may have acceleration errors, depending on the spacing of the actual encoder signals. Internally, this VI will govern the rate of the extrapolated crank angle ticks so that the sampling rate of the DAQ card is not exceeded. This feature is used when the crankshaft is accelerated and the extrapolated crank angle ticks are catching up to the current position at the rate of the 40MHz clock. The MaxSampleRate is the maximum sample rate of the DAQ card and is used to govern the rate of EncAB output. MaxSampleRate is configured as a ring control with 3 options of 250 KS/s, 500KS/s, or 1 MS/s. However, any value may be entered as a period in terms of 40MHz clock ticks. PulseBuffer is an indicator of how many EncAB pulses are currently being buffered for rate-governed output.



CurrentPosition (U16) The current position from the EPT.

EPTCamSig (Boolean) The Cam Signal from the EPT.

EPTCrankSig (Boolean) The Crank Signal from the EPT.

EncAB (Boolean) The encoder A or encoder B signals.

EncZ (Boolean) The encoder Z signal.

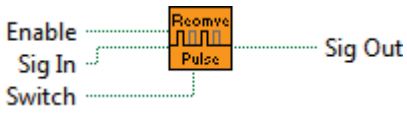
MaxSampleRate (U16) This control sets the maximum sample rate of the DAQ system.

PulseBuffer (U8) Displays the number of pulses buffered by the system.

SyncStopped (Boolean) The Sync Stopped from the EPT.

UseExtrap (Boolean) This control enables the encoder extrapolation.

dcat_remove_pulse (reentrant, subroutine) When enabled, every other pulse on input signal is removed.



Enable (Boolean) Enables the remove every other pulse.

Sig In (Boolean) Input signal.

Sig Out (Boolean) Output signal.

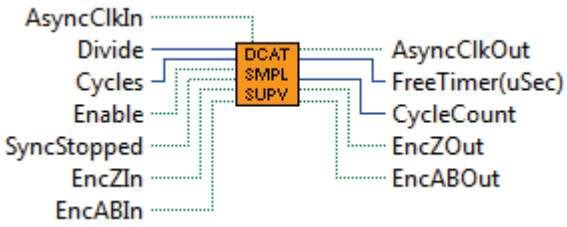
Switch (Boolean) Switches the pules that is removed.

dcat_sample_supv_reva (reentrant, normal priority) This is the DCAT Sampling Supervisor VI for LabVIEW FPGA. It accepts the high-resolution crankshaft encoder signal and encoder index from the EPT VI. The user may enable this VI at any time and the EncABOut and EncZOut outputs will be started and stopped at cycle boundaries to be used as a start trigger and sampling clock to the S-Series DAQ cards. If Cycles is 0 then the outputs will be started at the next engine cycle boundary (after enabled) and will continue until disabled. If Cycles is positive then the outputs will be started at the next engine cycle (after enabled) and continue for the specified number of engine cycles. The time-based AsyncClkIn is also gated on cycle boundaries to the AsyncClkOut output. This clock can be used for time-based sampling to an S-Series card for taking data simultaneously with angle-based data.

The Divide input can be used for dividing down the resolution of the EncABIn input. if Divide is 0 then EncABOut will remain FALSE. For passing the EncABIn through to EncABOut, set Divide to 1.

This VI also provides a FreeTimer in microseconds and a CycleCount. These values can be wired to the DCAT standard data VI for serializing this data to the S-Series card digital inputs.

The EncABOut and EncZOut outputs can be used as the ClockIn and StartTriggerIn inputs, respectively, to the DCAT standard data VI.



AsyncClkIn (Boolean)

AsyncClkOut (Boolean)

CycleCount (U32)

Cycles (U32)

Divide (U16)

Enable (Boolean)

EncABIn (Boolean)

EncABOut (Boolean)

EncZIn (Boolean)

EncZOut (Boolean)

FreeTimer(uSec) (U32) Returns the value of a free running counter at the time the VI wakes up. A free running counter rolls over when the counter reaches the maximum of <B>Size of Internal Counter</B> specified in the configuration dialog box.

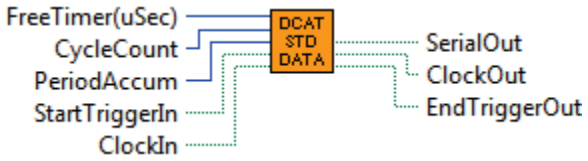
SyncStopped (Boolean)

dcat_std_data_reva (reentrant, normal priority) This VI accepts input values of FreeTimer (from DCAT Sample Supervisor), PeriodAccum (from the EPT) and CycleCount (from DCAT Sample Supervisor). These values are serialized with a UART protocol to be sampled by the S-Series digital input 0. This standard DCAT data is sampled along with angle-based and time-based data and decoded by real-time DCAT VIs. This data is stored with the analog data. This enables the engine speed, cycle count and FreeTimer to be known for each engine cycle.

The data values are serialed in the following order:
FreeTimer (uSec)
PeriodAccum
CycleCount

Each 32-bit word is encoded with 1 start bit (logic high) and 2 stop bits (logic low). The rising edge of the first start bit coincides with the rising edge of the first sample clock after the start trigger. Each data bit is encoded with 2 sample clocks.

The PeriodAccumCount input to the EPT VI should be set to the number of encoder pulses per crankshaft rotation. This enables the real-time DCAT software to calculate the correct engine speed for each engine cycle.



ClockIn (Boolean) Sample Clock In

ClockOut (Boolean) Sample Clock Out

CycleCount (U32) Cycle Counter

EndTriggerOut (Boolean) End Trigger Out

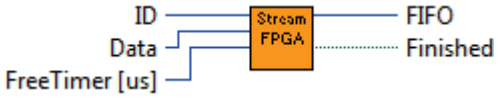
FreeTimer(uSec) (U32) FPGA Free Timmer

PeriodAccum (U32) Accumulated Cycle Period

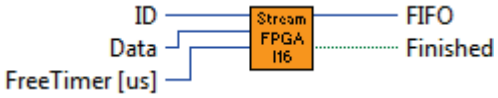
SerialOut (Boolean) Serial Data Line Out

StartTriggerIn (Boolean) Start Trigger In

dcat_stream_fpga_data (invalid priority) This VI converts unsigned integer data into a series of packets that can be placed in a FIFO and decoded by the real time system. The VI may be used in several parts of an application for different streams of data running at different sample rates. It should be placed in a while loop with the FIFO and the "Finished" terminal should be connected to the "Stop if True" condition of the while loop. This cannot be placed in a single cycle loop. The FIFO should be a U64 with enough resources to handle the desired throughput. The ID is used to differentiate between different streams of data. The timestamp is used to correlate data with the high speed data.



dcat_stream_fpga_data_i16 (reentrant, subroutine) This VI converts unsigned integer data into a series of packets that can be placed in a FIFO and decoded by the real time system. The VI may be used in several parts of an application for different streams of data running at different sample rates. It should be placed in a while loop with the FIFO and the "Finished" terminal should be connected to the "Stop if True" condition of the while loop. This cannot be placed in a single cycle loop. The FIFO should be a U64 with enough resources to handle the desired throughput. The ID is used to differentiate between different streams of data. The timestamp is used to correlate data with the high speed data.



Data (I16) FPGA Data

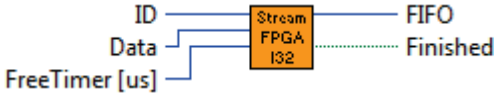
FIFO (U64) U64 Data for the FIFO

Finished (Boolean) Indicates the end of the data for the current measurement.

FreeTimer [us] (U32) FPGA Free Timer

ID (U8) Data Stream ID

dcat_stream_fpga_data_i32 (reentrant, subroutine) This VI converts unsigned integer data into a series of packets that can be placed in a FIFO and decoded by the real time system. The VI may be used in several parts of an application for different streams of data running at different sample rates. It should be placed in a while loop with the FIFO and the "Finished" terminal should be connected to the "Stop if True" condition of the while loop. This cannot be placed in a single cycle loop. The FIFO should be a U64 with enough resources to handle the desired throughput. The ID is used to differentiate between different streams of data. The timestamp is used to correlate data with the high speed data.



Data (I32) FPGA Data

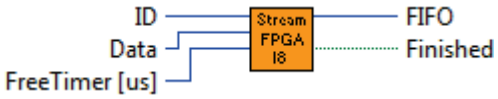
FIFO (U64) U64 Data for the FIFO

Finished (Boolean) Indicates the end of the data for the current measurement.

FreeTimer [us] (U32) FPGA Free Timer

ID (U8) Data Stream ID

dcat_stream_fpga_data_i8 (reentrant, subroutine) This VI converts unsigned integer data into a series of packets that can be placed in a FIFO and decoded by the real time system. The VI may be used in several parts of an application for different streams of data running at different sample rates. It should be placed in a while loop with the FIFO and the "Finished" terminal should be connected to the "Stop if True" condition of the while loop. This cannot be placed in a single cycle loop. The FIFO should be a U64 with enough resources to handle the desired throughput. The ID is used to differentiate between different streams of data. The timestamp is used to correlate data with the high speed data.



Data (I8) FPGA Data

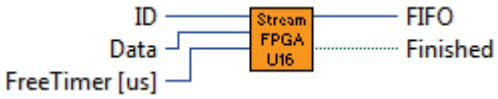
FIFO (U64) U64 Data for the FIFO

Finished (Boolean) Indicates the end of the data for the current measurement.

FreeTimer [us] (U32) FPGA Free Timer

ID (U8) Data Stream ID

dcat_stream_fpga_data_u16 (reentrant, subroutine) This VI converts unsigned integer data into a series of packets that can be placed in a FIFO and decoded by the real time system. The VI may be used in several parts of an application for different streams of data running at different sample rates. It should be placed in a while loop with the FIFO and the "Finished" terminal should be connected to the "Stop if True" condition of the while loop. This cannot be placed in a single cycle loop. The FIFO should be a U64 with enough resources to handle the desired throughput. The ID is used to differentiate between different streams of data. The timestamp is used to correlate data with the high speed data.



Data (U16) FPGA Data

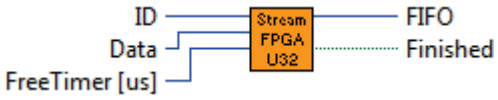
FIFO (U64) U64 Data for the FIFO

Finished (Boolean) Indicates the end of the data for the current measurement.

FreeTimer [us] (U32) FPGA Free Timer

ID (U8) Data Stream ID

dcat_stream_fpga_data_u32 (reentrant, subroutine) This VI converts unsigned integer data into a series of packets that can be placed in a FIFO and decoded by the real time system. The VI may be used in several parts of an application for different streams of data running at different sample rates. It should be placed in a while loop with the FIFO and the "Finished" terminal should be connected to the "Stop if True" condition of the while loop. This cannot be placed in a single cycle loop. The FIFO should be a U64 with enough resources to handle the desired throughput. The ID is used to differentiate between different streams of data. The timestamp is used to correlate data with the high speed data.



Data (U32) FPGA Data

FIFO (U64) U64 Data for the FIFO

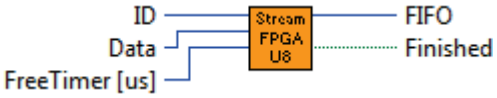
Finished (Boolean) Indicates the end of the data for the current measurement.

FreeTimer [us] (U32) FPGA Free Timer

ID (U8)

dcat_stream_fpga_data_u8 (reentrant, subroutine) This VI converts unsigned integer data into a series of packets that can be placed in a FIFO and decoded by the real time system. The VI may be used in several parts of an application for different streams of data running at different sample rates. It should be placed in a while loop with the FIFO and the "Finished" terminal should be connected to the "Stop if True" condition of the while loop. This cannot be

placed in a single cycle loop. The FIFO should be a U64 with enough resources to handle the desired throughput. The ID is used to differentiate between different streams of data. The timestamp is used to correlate data with the high speed data.



Data (U8) FPGA Data

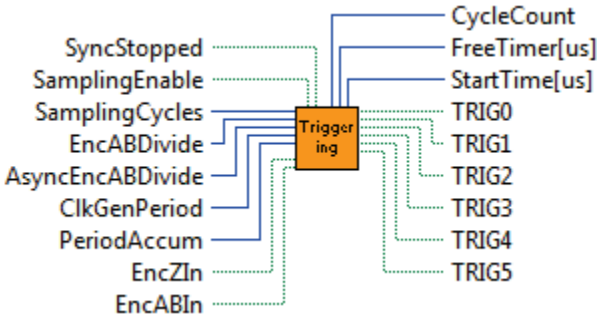
FIFO (U64) U64 Data for the FIFO

Finished (Boolean) Indicates the end of the data for the current measurement.

FreeTimer [us] (U32) FPGA Free Timer

ID (U8) Data Stream ID

dcat_triggering (reentrant, normal priority) This VI contains all of the triggering setup.



AsyncEncABDivide (U16) Value to divide the encoder A and encoder B by to reduce the encoder resolution to create the reduced resolution. A value of one should be used to leave the sample resolution alone.

ClkGenPeriod (U16) Asynchronous Clock Rate

CycleCount **CycleCount** (U32) Cylinder Count

EncABDivide (U16) Value to divide the encoder A and encoder B by to reduce the encoder resolution. A value of one should be used to leave the sample resolution alone.

EncABIn (Boolean) The encoder A or encoder B signal.

EncZIn (Boolean) The encoder Z signal.

FreeTimer[us] **FreeTimer[us]** (U32) FPGA Free Timer

PeriodAccum (U32) Cycle Period Time

SamplingCycles (U32) Specifies the number of cycles to sample.

SamplingEnable (Boolean) Enables the sampling.

StartTime[us] (U32) Sample start time.

SyncStopped (Boolean) The sync stopped signal from the EPT.

TRIG0 (Boolean) This control should be connected to the appropriate PXI trigger line.

TRIG1 (Boolean) This control should be connected to the appropriate PXI trigger line.

TRIG2 (Boolean) This control should be connected to the appropriate PXI trigger line.

TRIG3 (Boolean) This control should be connected to the appropriate PXI trigger line.

TRIG4 (Boolean) This control should be connected to the appropriate PXI trigger line.

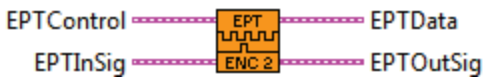
TRIG5 (Boolean) This control should be connected to the appropriate PXI trigger line.

ept_enc_vte2_dcat_revb (reentrant, normal priority) Engine Position Tracking VI

Pattern Type = Encoder

Extrapolation Level = 2 Bits.

To be placed within a single cycle loop of an FPGA block diagram, executed at 40 MHz.
See the Pattern-Specific Information section in the EPT User Manual.



EPTControl (Cluster) Cluster of control parameters for configuring engine position tracking for a specific pattern type.

EPTData (Cluster) Cluster of status information related to engine position tracking.

EPTInSig (Cluster) Cluster of crank and cam input signals used by the EPT for tracking engine position.

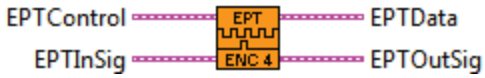
EPTOutSig (Cluster) Cluster of crank and cam outputs. These signals are connected directly to the inputs during normal operation, and connected to the simulated signals while SimEnable is TRUE.

ept_enc_vte4_dcat_revb (reentrant, normal priority) Engine Position Tracking VI

Pattern Type = Encoder

Extrapolation Level = 4 Bits.

To be placed within a single cycle loop of an FPGA block diagram, executed at 40 MHz.
See the Pattern-Specific Information section in the EPT User Manual.



EPTControl (Cluster) Cluster of control parameters for configuring engine position tracking for a specific pattern type.

EPTData (Cluster) Cluster of status information related to engine position tracking.

EPTInSig (Cluster) Cluster of crank and cam input signals used by the EPT for tracking engine position.

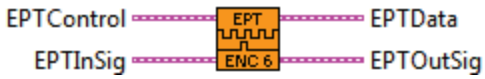
EPTOutSig (Cluster) Cluster of crank and cam outputs. These signals are connected directly to the inputs during normal operation, and connected to the simulated signals while SimEnable is TRUE.

ept_enc_vte6_dcat_revb (reentrant, normal priority) Engine Position Tracking VI

Pattern Type = Encoder

Extrapolation Level = 6 Bits.

To be placed within a single cycle loop of an FPGA block diagram, executed at 40 MHz.
See the Pattern-Specific Information section in the EPT User Manual.



EPTControl (Cluster) Cluster of control parameters for configuring engine position tracking for a specific pattern type.

EPTData (Cluster) Cluster of status information related to engine position tracking.

EPTInSig (Cluster) Cluster of crank and cam input signals used by the EPT for tracking engine position.

EPTOutSig (Cluster) Cluster of crank and cam outputs. These signals are connected directly to the inputs during normal operation, and connected to the simulated signals while SimEnable is TRUE.

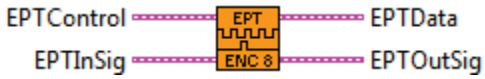
ept_enc_vte8_dcat_revb (reentrant, normal priority) Engine Position Tracking VI

Pattern Type = Encoder

Extrapolation Level = 8 Bits.

To be placed within a single cycle loop of an FPGA block diagram, executed at 40 MHz.

See the Pattern-Specific Information section in the EPT User Manual.



EPTControl (Cluster) Cluster of control parameters for configuring engine position tracking for a specific pattern type.

EPTData (Cluster) Cluster of status information related to engine position tracking.

EPTInSig (Cluster) Cluster of crank and cam input signals used by the EPT for tracking engine position.

EPTOutSig (Cluster) Cluster of crank and cam outputs. These signals are connected directly to the inputs during normal operation, and connected to the simulated signals while SimEnable is TRUE.

Appendix B – References

- Cengel, Y. A., Boles, M. A., 2002, “Thermodynamics: An Engineering Approach”, Fourth Edition, McGraw Hill, Inc., Boston, MA
- Gatowski, J. A., Balles, E. N., Chun, K. M., Nelson, F. E., Ekchian, J. A., Heywood, J. B., 1984, “Heat Release Analysis of Engine Pressure Data”, SAE Technical Paper Series 841359
- Grimm, B. M., Johnson, R. T., 1990, “Review of Simple Heat Release Computations”, SAE Technical Paper Series 900445
- Hayes, T. K., Savage, L. D., Sorenson, S. C., 1986, “Cylinder Pressure Data Acquisition and Heat Release Analysis on a Personal Computer”, SAE Technical Paper Series 860029
- Herbert, A. J., Ressell, M. F., 1982, “Measurement of Combustion Noise in Diesel Engines”, Mechanical Engineering Technology
- Heywood, J. B., 1988, “Internal Combustion Engine Fundamentals”, McGraw Hill, Inc., New York, NY
- Sellnau, M. C., Matekunas, F. A., Battiston, P. A., Chang, C. F., Lancaster, D. R., 2000, “Cylinder-Pressure-Based engine Control Using Pressure-Ratio-Management and Low-Cost Non-Intrusive Cylinder Pressure Sensors”, SAE Technical Paper Series 2000-01-0932
- Shelby, M. H., Stein, R. A., Warren, C. C., 2003, “A New Analysis Method for Accurate Accounting of SI Engine Pumping and Indicated Work”, SAE International 04ANNUAL-411
- Stone, R., 1999, “Introduction to Internal Combustion Engines”, Third Edition, Society of Automotive Engineers, Inc., Warrendale, PA